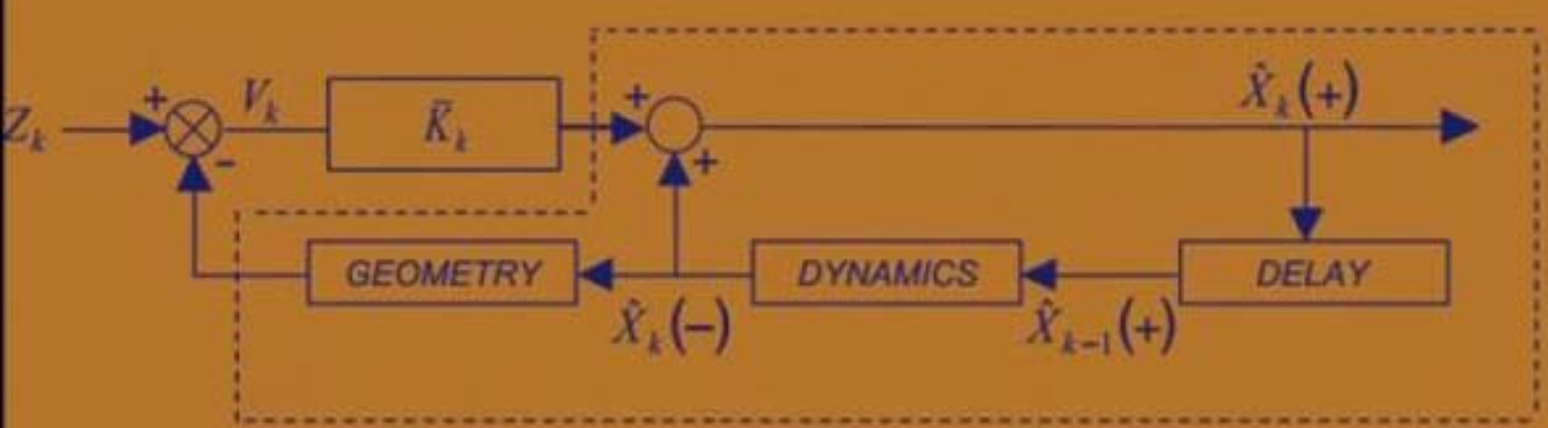
CD-ROM INCLUDED

Kalman Filtering

Theory and Practice Using MATLAB®

Third Edition

Mohinder S. Grewal
Angus P. Andrews



KALMAN FILTERING

KALMAN FILTERING

Theory and Practice Using MATLAB®

Third Edition

MOHINDER S. GREWAL
California State University at Fullerton

ANGUS P. ANDREWS
Rockwell Science Center (retired)

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:

Grewal, Mohinder S.
Kalman filtering : theory and practice using MATLAB/Mohinder S. Grewal, Angus P. Andrews. — 3rd ed.
p. cm.
Includes bibliographical references and index.
ISBN 978-0-470-17366-4 (cloth)
1. Kalman filtering. 2. MATLAB. I. Andrews, Angus P. II. Title.
QA402.3.G695 2008
629.8'312—dc22

200803733

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

CONTENTS

PREFACE

This book is designed to provide familiarity with both the *theoretical* and *practical* aspects of Kalman filtering by including real-world problems in practice as illustrative examples. The material includes the essential technical background for Kalman filtering and the more practical aspects of implementation: how to represent the problem in a mathematical model, analyze the performance of the estimator as a function of system design parameters, implement the mechanization equations in numerically stable algorithms, assess its computational requirements, test the validity of results, and monitor the filter performance in operation. These are important attributes of the subject that are often overlooked in theoretical treatments but are necessary for application of the theory to real-world problems.

In this third edition, we have included important developments in the implementation and application of Kalman filtering over the past several years, including adaptations for nonlinear filtering, more robust smoothing methods, and developing applications in navigation.

We have also incorporated many helpful corrections and suggestions from our readers, reviewers, colleagues, and students over the past several years for the overall improvement of the textbook.

All software has been provided in MATLAB[1] so that users can take advantage of its excellent graphing capabilities and a programming interface that is very close to the mathematical equations used for defining Kalman filtering and its applications. See Appendix A for more information on MATLAB software.

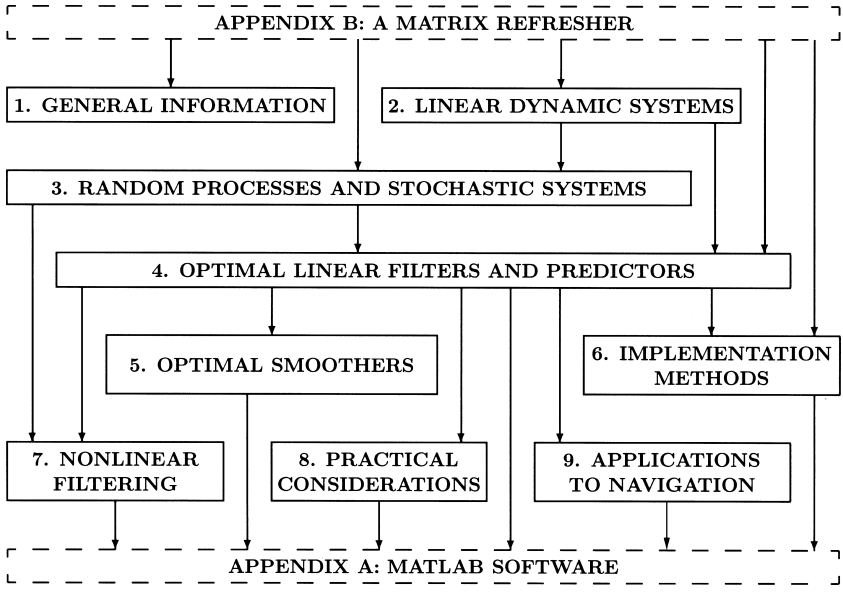
The inclusion of the software is practically a matter of necessity, because Kalman filtering would not be very useful without computers to implement it. It provides a

[1]MATLAB is a registered trademark of The Mathworks, Inc.

better learning experience for the student to discover how the Kalman filter works by observing it in action.

The implementation of Kalman filtering on computers also illuminates some of the practical considerations of finite-wordlength arithmetic and the need for alternative algorithms to preserve the accuracy of the results. If the student wishes to apply what she or he learns, then it is essential that she or he experience its workings and failings—and learn to recognize the difference.

The book is organized as a text for an introductory course in stochastic processes at the senior level and as a first-year graduate-level course in Kalman filtering theory and application. It can also be used for self-instruction or for purposes of review by practicing engineers and scientists who are not intimately familiar with the subject. The organization of the material is illustrated by the following chapter-level dependency graph, which shows how the subject of each chapter depends upon material in other chapters. The arrows in the figure indicate the recommended order of study. Boxes above another box and connected by arrows indicate that the material represented by the upper boxes is background material for the subject in the lower box.



Chapter 1 provides an informal introduction to the general subject matter by way of its history of development and application. Chapters 2 and 3 and Appendix B cover the essential background material on linear systems, probability, stochastic processes, and modeling. These chapters could be covered in a senior-level course in electrical, computer, and systems engineering.

Chapter 4 covers linear optimal filters and predictors, with detailed examples of applications. Chapter 5 is a new tutorial-level treatment of optimal smoothing

methods based on Kalman filtering models, including more recent, more robust implementations. Chapter 7 is devoted to nonlinear applications, including extended Kalman filters for quasilinear problems, and to sampling-based methods for extending Kalman filtering to more highly nonlinear problems. Applications of these techniques to the identification of unknown parameters of systems are given as examples. Chapter 6 covers the more modern implementation techniques, with algorithms provided for computer implementation.

Chapter 8 deals with more practical matters of implementation and use beyond the numerical methods of Chapter 7. These matters include memory and throughput requirements (and methods to reduce them), divergence problems (and effective remedies), and practical approaches to suboptimal filtering and measurement selection.

As a demonstration of how to develop and evaluate applications of Kalman filtering, in Chapter 9 we show how to develop different Kalman filtering configurations for integrating global navigation satellite system receivers with inertial navigation systems.

Chapters 4–8 cover the essential material for a first-year graduate class in Kalman filtering theory and application or as a basic course in digital estimation theory and application.

MOHINDER S. GREWAL, PHD, PE

ANGUS P. ANDREWS, PHD

ACKNOWLEDGMENTS

The authors express their appreciation to the following individuals for their contributions during the preparation of the core material: Robert W. Bass, E. Richard Cohen, Thomas W. De Vries, Reverend Joseph Gaffney, Thomas L. Gunckel II, Dwayne Heckman, Robert A. Hubbs, Thomas Kailath, Rudolf E. Kalman, Alan J. Laub, Robert F. Nease, John C. Pinson, John M. Richardson, Jorma Rissanen, Gerald E. Runyon, Joseph Smith, and Donald F. Wiberg. We also express our appreciation to Donald Knuth and Leslie Lamport for TEX and LATEX, respectively.

In addition, the following individuals deserve special recognition for their careful review, corrections, and suggestions for improving the second edition: Dean Dang and Gordon Inverarity.

For the third edition, we wish to thank Kenneth W. Fertig, former Chief Statistician of Rockwell International, for providing us with better statistical examples. We also thank Mark Arlinghaus and Seung Hyun Kong for needed corrections to the second edition.

Most of all, for their dedication, support, and understanding throughout the writing of all editions, we dedicate this book to Sonja Grewal and Jeri Andrews.

MOHINDER S. GREWAL

ANGUS P. ANDREWS

LIST OF ABBREVIATIONS

ANSI, American National Standards Institute

arc-sec, second of arc

BMFLS, Biswas-Mahalanabis fixed-lag smoother

CEP, circular error probable, the radius of a circle centered at the mean of a probability distribution such that is equally likely that a random sample is inside or outside the circle (also called *circle of equal probability*)

dB, decibel

ed., editor or edition

EKF, extended Kalman filter

ENU, east-north-up (coordinates)

f, foot (0.3048 m)

flops, floating-point operations per second

FLS, fixed lag smoother

g, 9.80665 m/s^2

GNSS, global navigation satellite system

GPS, Global Positioning Service, a GNSS operated by the U.S. Department of Defense

h, hour

IEEE, Institute of Electrical and Electronic Engineers

IEKF, iterated extended Kalman filter

IIR, infinite impulse response

INS, inertial navigation system

ISA, inertial sensor assembly
KF, Kalman filter
km, kilometer
kph, kilometer per hour
m, meter
max, maximum
mi, mile
min, minute of time, or minimum
mph, mile per hour
NED, north-east-down (coordinates)
NMi, nautical mile (1852 m)
ppm, part per million
PSD, power spectral density
RMS, root mean-squared
RP, random process
RPY, roll-pitch-yaw (vehicle coordinates)
RS, random sequence
RV, random variable
s, second of time
SKF, Schmidt–Kalman filter
SPKF, sigma-point Kalman filter
STM, state transition matrix
SVD, singular value decomposition
UKF, unscented Kalman filter
UT, unscented transform
vs, versus
WSS, wide-sense stationary
μ, micron (10^{-6} m) or micro (10^{-6} [units])

1

GENERAL INFORMATION

. . . the things of this world cannot be made known without mathematics.
—Roger Bacon (1220–1292), *Opus Majus*, trans. R. Burke, 1928

1.1 ON KALMAN FILTERING

1.1.1 First of All: What Is a Kalman Filter?

Theoretically, the Kalman filter is an estimator for what is called the *linear-quadratic problem*, which is the problem of estimating the instantaneous "state" (a concept that will be made more precise in the next chapter) of a linear dynamic system perturbed by white noise—by using measurements linearly related to the state but corrupted by white noise. The resulting estimator is statistically optimal with respect to any quadratic function of estimation error.

Practically, the Kalman filter is one of the greater discoveries in the history of statistical estimation theory and possibly the greatest discovery in the twentieth century. It has enabled humankind to do many things that could not have been done without it, and it has become as indispensable as silicon in the makeup of many electronic systems. Its most immediate applications have been for the control of complex dynamic systems such as continuous manufacturing processes, aircraft, ships, or spacecraft. To control a dynamic system, you must first know what it is doing. For these applications, it is not always possible or desirable to measure every variable

Kalman Filtering: Theory and Practice using MATLAB®, Third Edition. By Mohinder S. Grewal and Angus P. Andrews

that you want to control, and the Kalman filter provides a means for inferring the missing information from indirect and noisy measurements. The Kalman filter is also used for predicting the likely future courses of dynamic systems that people are not likely to control, such as the flow of rivers during floods, the trajectories of celestial bodies, or the prices of traded commodities.

From a practical standpoint, these are the perspectives that this book will present:

- *It is only a tool.* It does not solve any problem all by itself, although it can make it easier for you to do so. It is not a *physical* tool but a *mathematical* one. Mathematical tools make mental work more efficient, just as mechanical tools make physical work more efficient. As with any tool, it is important to understand its use and function before you can apply it effectively. The purpose of this book is to make you sufficiently familiar with and proficient in the use of the Kalman filter so that you can apply it correctly and efficiently.
- *It is a computer program.* It has been called "ideally suited to digital computer implementation" [95], in part because it uses a *finite representation* of the estimation problem—by a *finite* number of variables. It does, however, assume that these variables are *real numbers*—with *infinite* precision. Some of the problems encountered in its use arise from the distinction between finite dimension and finite information and from the distinction between finite and manageable problem sizes. These are all issues on the practical side of Kalman filtering that must be considered along with the theory.
- *It is a consistent statistical characterization of an estimation problem.* It is much more than an *estimator*, because it propagates the current *state of knowledge* of the dynamic system, including the statistical influence of random dynamic perturbations and the effects of all past measurements. These properties are extremely useful for statistical analysis and the predictive design of sensor systems.

If these answers provide the level of understanding that you are seeking, then there is no need for you to read the rest of this book. If you need to understand Kalman filters well enough to use them effectively, then please read on!

1.1.2 How It Came to Be Called a Filter

It might seem strange that the term *filter* would apply to an estimator. More commonly, a filter is a physical device for removing unwanted fractions of mixtures. (The word *felt* comes from the same medieval Latin stem, for the material was used as a filter for liquids.) Originally, a filter solved the problem of separating unwanted components of gas-liquid-solid mixtures. In the era of crystal radios and vacuum tubes, the term was applied to analog circuits that filter electronic signals. These signals are mixtures of different frequency components, and these physical devices preferentially attenuate unwanted frequencies.

This concept was extended in the 1930s and 1940s to the separation of signals from noise, both of which were characterized by their power spectral densities.

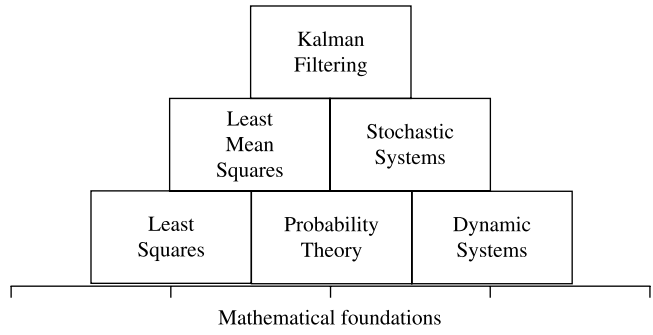


Figure 1.1 Foundational concepts in Kalman filtering.

Kolmogorov and Wiener used this statistical characterization of their probability distributions in forming an optimal estimate of the signal, given the sum of the signal and noise.

With Kalman filtering, the term *filter* assumed a meaning that is well beyond the original idea of *separation* of the components of a mixture. It has also come to include the solution of an *inversion problem*, in which one knows how to represent the measurable variables as functions of the variables of principal interest. In essence, it inverts this functional relationship and estimates the independent variables as inverted functions of the dependent (measurable) variables. These variables of interest are also allowed to be dynamic, with dynamics that are only partially predictable.

1.1.3 Its Mathematical Foundations

Figure 1.1 depicts the essential subjects forming the foundations for Kalman filtering theory. Although it shows Kalman filtering as the apex of a pyramid, it is itself but part of the foundations of another discipline—modern control theory—and a proper subset of statistical decision theory.

We will examine only the top three layers of the pyramid in this book and a little of the underlying mathematics[1] (matrix theory) in Appendix B.

1.1.4 What It Is Used For

The applications of Kalman filtering encompass many fields, but its use as a tool is almost exclusively for two purposes: *estimation* and *performance analysis* of estimators.

1. **Estimating the state of dynamic systems.** What is a dynamic system? Almost everything, if you are picky about it. Except for a few fundamental physical

[1]It is best that one not examine the lowest layers of these mathematical foundations too carefully anyway. They eventually rest on human intellect, the foundations of which are not as well understood.

TABLE 1.1 Examples of Estimation Problems

Application	Dynamic System	Sensor Types
Process control	Chemical plant	Pressure Temperature Flow rate Gas analyzer
Flood prediction	River system	Water level Rain gauge Weather radar
Tracking	Spacecraft	Radar Imaging system
Navigation	Ship	Sextant Log Gyroscope Accelerometer GNSS* receiver

*Global navigation satellite system.

constants, there is hardly anything in the universe that is truly *constant*. The orbital parameters of the dwarf planet Ceres are not constant, and even the "fixed" stars and continents are moving. Nearly all physical systems are dynamic to some degree. If one wants very precise estimates of their characteristics over time, then one has to take their dynamics into consideration. The problem is that one does not always know their dynamics very precisely either. Given this state of partial ignorance, the best one can do is express our ignorance more precisely—using probabilities. The Kalman filter allows us to estimate the state of dynamic systems with certain types of random behavior by using such statistical information. A few examples of such systems are listed in the second column of Table 1.1.

2. **Performance analysis of estimation systems.** The third column of Table 1.1 lists some possible sensor types that might be used in estimating the state of the corresponding dynamic systems. The objective of design analysis is to determine how best to use these sensor types for a given set of design criteria. These criteria are typically related to estimation accuracy and system cost.

The Kalman filter uses a parametric characterization of the probability distribution of its estimation errors in determining the optimal filtering gains, and this probability distribution may be used in assessing its performance as a function of the "design parameters" of an estimation system, such as

- the types of sensors to be used,
- the locations and orientations of the various sensor types with respect to the system to be estimated,
- the allowable noise characteristics of the sensors,

- the prefiltering methods for smoothing sensor noise,
- the data sampling rates for the various sensor types, and
- the level of model simplification to reduce implementation requirements.

The analytical capability of Kalman filter formalism also allows a system designer to assign an "error budget" to subsystems of an estimation system and to trade off the budget allocations to optimize cost or other measures of performance while achieving a required level of estimation accuracy.

1.2 ON OPTIMAL ESTIMATION METHODS

The Kalman filter is the result of an evolutionary process of ideas from many creative thinkers over many centuries. We present here some of the seminal ideas in this process, the discoverers of which are listed in historical perspective in Figure 1.2. This list is by no means exhaustive. There are far too many people involved to show them all, but the figure should give some idea of the time periods involved. The figure covers only half a millennium, and the study and development of mathematical concepts goes back beyond history. Readers interested in more detailed histories of optimal estimation are referred to the survey articles by Kailath [133, 130], Lainiotis [160], Mendel and Gieseking [194], and Sorenson [254, 256] and the personal accounts of Battin [25] and Schmidt [241].

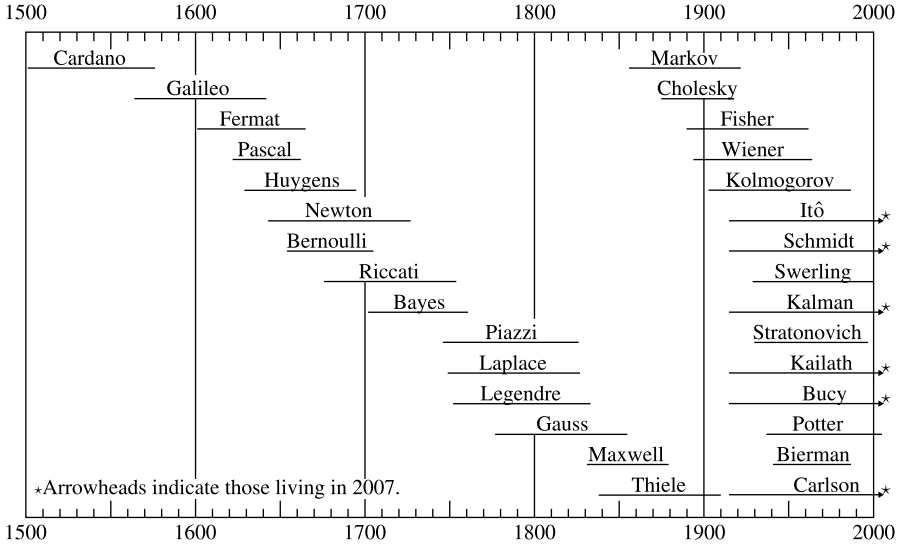


Figure 1.2 Lifelines of some important contributors to estimation technology.

1.2.1 Beginnings of Optimal Estimation Theory

The first method for forming an *optimal* estimate from noisy data is the *method of least squares*. Its discovery is generally attributed to Carl Friedrich Gauss (1777–1855) in 1795. The inevitability of measurement errors had been recognized since the time of Galileo Galilei (1564–1642), but this was the first formal method for dealing with them. Although it is more commonly used for linear estimation problems, Gauss first used it for a nonlinear estimation problem in mathematical astronomy, which was part of an interesting event in the history of astronomy. The following account was put together from several sources, including the account by Makemson in [19].

On January 1, 1801, the first day of the nineteenth century, the Italian astronomer Giuseppe Piazzi was checking an entry in a star catalog. Unbeknown to Piazzi, it included an error by the printer. While searching for the "missing" star, Piazzi discovered instead something that moved. It was the dwarf planet *Ceres*—the largest body in the asteroid belt and the first to be discovered—but Piazzi did not know that yet. He was able to track and measure its apparent motion against the "fixed" star background during 41 nights before it moved too close to the sun and disappeared.

On January 24, Piazzi had written of his discovery to Johann Bode. Bode is best known for *Bode's law*, which states that the distances of the planets from the sun, in astronomical units, are given by the sequence

$$d_n = \frac{1}{10}(4 + 3 \times 2^n) \quad \text{for } n = -\infty, 0, 1, 2, ?, 4, 5, \ldots. \tag{1.1}$$

Actually, it was not Bode but Johann Tietz who first proposed this formula, in 1772. At that time there were only six known planets. In 1781, Friedrich Herschel discovered Uranus, which fit nicely into this formula for $n = 6$. No planet had been discovered for $n = 3$. Spurred on by Bode, an association of European astronomers had been searching for the "missing" eighth planet for nearly 30 years. Piazzi was not part of this association, but he did inform Bode of his unintended discovery.

Piazzi's letter did not reach Bode until March 20. (Electronic mail was discovered much later.) Bode suspected that Piazzi's discovery might be the missing planet, but there was insufficient data for determining its orbital elements by the methods then available. It is a problem in nonlinear equations that Newton himself had declared as being among the most difficult in mathematical astronomy. Nobody had solved it and, as a result, Ceres was lost in space again.

Piazzi's discoveries were not published until the autumn of 1801. The possible discovery—and subsequent loss—of a new planet, coinciding with the beginning of a new century, was exciting news. It contradicted a philosophical justification for the existence of only seven planets—the number known before Ceres and a number defended by the respected philosopher Georg Hegel, among others. Hegel had recently published a book in which he chastised the astronomers for wasting their time searching for an eighth planet when there was sound philosophical justification for there being only seven. The new celestial object became a subject of

conversation in intellectual circles nearly everywhere. Fortunately, the problem caught the attention of a 24-year-old mathematician at Göttingen named Carl Friedrich Gauss.

Gauss had toyed with the orbit determination problem a few weeks earlier but had set it aside for other interests. He now devoted most of his time to the problem, produced an estimate of the orbit of Ceres in December, and sent his results to Piazzi. The new "planet" (later reclassified as an asteroid), which had been sighted on the first day of the year, was found again—by its discoverer—on the last day of the year.

Gauss did not publish his orbit determination methods until 1809.[2] In this publication, he also described the method of least squares that he had discovered in 1795, at the age of 18, and had used in refining his estimates of the orbit of Ceres.

Although Ceres played a significant role in the history of discovery and still reappears regularly in the nighttime sky, it had faded into obscurity as an object of intellectual interest until the 2007 launch of scientific probe Dawn for a 2015 rendezvous with Ceres. The method of least squares, on the other hand, has been an object of continuing interest and benefit to generations of scientists and technologists ever since its introduction. It has had a profound effect on the history of science. It was the first optimal estimation method, and it provided an important connection between the experimental and theoretical sciences: It gave experimentalists a practical method for estimating the unknown parameters of theoretical models.

1.2.2 Method of Least Squares

The following example of a least-squares problem is the one most often seen, although the *method* of least squares may be applied to a much greater range of problems.

Example 1.1 (Least-Squares Solution for Overdetermined Linear Systems) Gauss discovered that if he wrote a system of equations in matrix form, as

$$\begin{bmatrix} h_{11} & h_{12} & h_{13} & \dots & h_{1n} \\ h_{21} & h_{22} & h_{23} & \dots & h_{2n} \\ h_{31} & h_{32} & h_{33} & \dots & h_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ h_{l1} & h_{l2} & h_{l3} & \dots & h_{ln} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ \vdots \\ z_l \end{bmatrix} \tag{1.2}$$

[2]In the meantime, the method of least squares had been discovered independently and published by Andrien-Marie Legendre (1752–1833) in France and Robert Adrian (1775–1855) in the United States [130]. [It had also been discovered and used before Gauss was born by the German–Swiss physicist Johann Heinrich Lambert (1728–1777).] Such Jungian synchronicity (i.e., the phenomenon of multiple near-simultaneous discovery) was to be repeated for other breakthroughs in estimation theory, as well—for the Wiener–Kolmogorov filter and the Kalman filter.

or

$$Hx = z, \tag{1.3}$$

then he could consider the problem of solving for that value of an estimate $\hat{x}$ (pronounced "x-hat") that minimizes the "estimated measurement error" $H\hat{x} - z$. He could characterize that estimation error in terms of its Euclidean vector norm $|H\hat{x} - z|$, or, equivalently, its square:

$$\varepsilon^2(\hat{x}) = |H\hat{x} - z|^2 \tag{1.4}$$

$$= \sum_{i=1}^{m} \left[\sum_{j=1}^{n} h_{ij}\hat{x}_j - z_i \right]^2, \tag{1.5}$$

which is a continuously differentiable function of the n unknowns $\hat{x}_1, \hat{x}_2, \hat{x}_3, \ldots, \hat{x}_n$. This function $\varepsilon^2(\hat{x}) \to \infty$ as any component $\hat{x}_k \to \pm\infty$. Consequently, it will achieve its minimum value where all its derivatives with respect to the $\hat{x}_k$ are zero. There are n such equations of the form

$$0 = \frac{\partial \varepsilon^2}{\partial \hat{x}_k} \tag{1.6}$$

$$= 2 \sum_{i=1}^{m} h_{ik} \left[\sum_{j=1}^{n} h_{ij}\hat{x}_j - z_i \right] \tag{1.7}$$

for $k = 1, 2, 3, \ldots, n$. Note that in this last equation the expression

$$\sum_{j=1}^{n} h_{ij}\hat{x}_j - z_i = \{H\hat{x} - z\}_i, \tag{1.8}$$

the ith row of $H\hat{x} - z$, and the outermost summation is equivalent to the dot product of the kth column of H with $H\hat{x} - z$. Therefore, Equation 1.7 can be written as

$$0 = 2H^{\mathrm{T}}[H\hat{x} - z] \tag{1.9}$$

$$= 2H^{\mathrm{T}}H\hat{x} - 2H^{\mathrm{T}}z \tag{1.10}$$

or

$$H^{\mathrm{T}}H\hat{x} = H^{\mathrm{T}}z,$$

where the matrix transpose H^{T} is defined as

$$H^{\mathrm{T}} = \begin{bmatrix} h_{11} & h_{21} & h_{31} & \dots & h_{m1} \\ h_{12} & h_{22} & h_{32} & \dots & h_{m2} \\ h_{13} & h_{23} & h_{33} & \dots & h_{m3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ h_{1n} & h_{2n} & h_{3n} & \dots & h_{mn} \end{bmatrix} \tag{1.11}$$

the normal equation of the linear least squares problem. The equation

$$H^{\mathrm{T}} H\hat{x} = H^{\mathrm{T}} z \tag{1.12}$$

is called the *normal equation* or the *normal form of the equation for the linear least-squares problem*. It has precisely as many equivalent scalar equations as unknowns.

The Gramian of the Linear Least Squares Problem The normal equation has the solution

$$\hat{x} = (H^{\mathrm{T}} H)^{-1} H^{\mathrm{T}} z,$$

provided that the matrix

$$\mathcal{G} = H^{\mathrm{T}} H \tag{1.13}$$

is *nonsingular* (i.e., invertible). The matrix product $\mathcal{G} = H^T H$ in this equation is called the *Gramian matrix*.[3] The determinant of the Gramian matrix characterizes whether or not the column vectors of H are linearly independent. If its determinant is zero, the column vectors of H are linearly dependent and $\hat{x}$ cannot be determind uniquely. If its determinant is nonzero, then the solution $\hat{x}$ is uniquely determined.

Least-Squares Solution In the case where the Gramian matrix *is* invertible (i.e., nonsingular), the solution $\hat{x}$ is called the *least-squares solution of the overdetermined linear inversion problem*. It is an estimate that makes no assumptions about the

[3]Named for the Danish mathematician Jørgen Pedersen Gram (1850–1916). This matrix is also related to what is called the *unscaled Fisher information matrix*, named after the English statistician Ronald Aylmer Fisher (1890–1962). Although information matrices and Gramian matrices have different definitions and uses, they can amount to almost the same thing in this particular instance. The formal statistical definition of the term *information matrix* represents the information obtained from a sample of values from a known probability distribution. It corresponds to a scaled version of the Gramian matrix when the measurement errors in z have a joint Gaussian distribution, with the scaling related to the uncertainty of the measured data. The information matrix is a *quantitative* statistical characterization of the "information" (in some sense) that is in the data z used for estimating x. The Gramian, on the other hand, is used as an *qualitative* algebraic characterization of the uniqueness of the solution.

nature of the unknown measurement errors, although Gauss alluded to that possibility in his description of the method. The formal treatment of uncertainty in estimation would come later.

This form of the Gramian matrix will be used in Chapter 2 to define the observability matrix of a linear dynamic system model in discrete time.

1.2.2.1 Least Squares in Continuous Time The following example illustrates how the principle of least squares can be applied to fitting a vector-valued parametric model to data in continuous time. It also illustrates how the issue of *determinacy* (i.e., whether there is a *unique* solution to the problem) is characterized by the Gramian matrix in this context.

Example 1.2 (Least-Squares Fitting of Vector-Valued Data in Continuous Time) Suppose that, for each value of time t on an interval, $t_0 \le t \le t_f$, $z(t)$ is an ℓ-dimensional signal vector that is modeled as a function of an unknown n-vector x by the equation

$$z(t) = H(t)\,x,$$

where $H(t)$ is a known $\ell \times n$ matrix. The squared error in this relation at each time t will be

$$\begin{aligned}\varepsilon^2(t) &= |z(t) - H(t)\,\hat{x}|^2 \\ &= \hat{x}^{\mathrm{T}}[H^{\mathrm{T}}(t)H(t)]\hat{x} - 2\hat{x}^{\mathrm{T}}H^{\mathrm{T}}(t)z(t) + |z(t)|^2.\end{aligned}$$

The squared integrated error over the interval will then be the integral

$$\begin{aligned}\|\varepsilon\|^2 &= \int_{t_0}^{t_f} \varepsilon^2(t)\,t \\ &= \hat{x}^{\mathrm{T}}\left[\int_{t_0}^{t_f} H^{\mathrm{T}}(t)H(t)\,dt\right]\hat{x} - 2\hat{x}^{\mathrm{T}}\left[\int_{t_0}^{t_f} H^{\mathrm{T}}(t)z(t)\,dt\right] + \int_{t_0}^{t_f} |z(t)|^2\,dt,\end{aligned}$$

which has exactly the same array structure with respect to $\hat{x}$ as the algebraic least-squares problem. The least-squares solution for $\hat{x}$ can be found, as before, by taking the derivatives of $\|\varepsilon\|^2$ with respect to the components of $\hat{x}$ and equating them to zero. The resulting equations have the solution

$$\hat{x} = \left[\int_{t_0}^{t_f} H^{\mathrm{T}}(t)H(t)\,dt\right]^{-1}\left[\int_{t_0}^{t_f} H^{\mathrm{T}}(t)\,z(t)\,dt\right],$$

provided that the corresponding Gramian matrix

$$G = \int_{t_0}^{t_f} H^{\mathrm{T}}(t)H(t)\,dt$$

is nonsingular.

This form of the Gramian matrix will be used in Chapter 2 to define the observability matrix of a linear dynamic system model in continuous time.

1.2.3 Gramian Matrix and Observability

For the examples considered above, observability does not depend upon the measurable data (z). It depends only on the nonsingularity of the Gramian matrix (G), which depends only on the linear constraint matrix (H) between the unknowns and knowns.

Observability of a set of unknown variables is the issue of whether or not their values are *uniquely determinable* from a given set of *constraints*, expressed as equations involving functions of the unknown variables. The unknown variables are said to be *observable* if their values are uniquely determinable from the given constraints, and they are said to be *unobservable* if they are not uniquely determinable from the given constraints.

The condition of *nonsingularity* (or *full rank*) of the Gramian matrix is an *algebraic* characterization of observability when the constraining equations are *linear* in the unknown variables. It also applies to the case where the constraining equations are not exact due to errors in the values of the allegedly known parameters of the equations.

The Gramian matrix will be used in Chapter 2 to define observability of the states of dynamic systems in continuous time and discrete time.

1.2.4 Mathematical Modeling of Uncertainty

Probabilities represent the state of knowledge about physical phenomena by providing something more useful than "I don't know" to questions involving uncertainty. One of the mysteries in the history of science is why it took so long for mathematicians to formalize a subject of such practical importance. The Romans were selling insurance and annuities long before expectancy and risk were concepts of serious mathematical interest. Much later, the Italians were issuing insurance policies against business risks in the early Renaissance, and the first known attempts at a theory of probabilities—for games of chance—occurred in that period. The Italian Girolamo Cardano[4] (1501–1576) performed an accurate analysis of probabilities

[4]Cardano was a practicing physician in Milan who also wrote books on mathematics. His book *De Ludo Hleae*, on the mathematical analysis of games of chance (principally dice games), was published nearly a century after his death. Cardano was also the inventor of the most common type of universal joint found in automobiles, sometimes called the *Cardan joint* or *Cardan shaft*.

for games involving dice. He assumed that successive tosses of the dice were statistically independent events. Like the pioneering Indian mathematician Brahmagupta (589–668), Cardano stated without proof that the accuracies of empirical statistics tend to improve with the number of trials. This would later be formalized as a *law of large numbers.*

More general treatments of probabilities were developed by Blaise Pascal (1622–1662), Pierre de Fermat (1601–1655), and Christiaan Huygens (1629–1695). Fermat's work on combinations was taken up by Jakob (or James) Bernoulli (1654–1705), who is considered by some historians to be the founder of probability theory. He gave the first rigorous proof of the law of large numbers for repeated independent trials (now called *Bernoulli trials*). Thomas Bayes (1702–1761) derived his famous rule for statistical inference sometime after Bernoulli. Abraham de Moivre (1667–1754), Pierre Simon Marquis de Laplace (1749–1827), Adrien Marie Legendre (1752–1833), and Carl Friedrich Gauss (1777–1855) continued this development into the nineteenth century.

Between the early nineteenth century and the mid-twentieth century, the probabilities themselves began to take on more meaning as physically significant attributes. The idea that the laws of nature embrace random phenomena and that these are treatable by probabilistic models began to emerge in the nineteenth century. The development and application of probabilistic models for the physical world expanded rapidly in that period. It even became an important part of sociology. The work of James Clerk Maxwell (1831–1879) in statistical mechanics established the probabilistic treatment of natural phenomena as a scientific (and successful) discipline. Andrei Andreyevich Markov (1856–1922) would develop much of the theory of what is today called a *Markov process* (in continuous time) or *Markov chain* (in discrete time), a random process with the property that the evolution over time of its probability distribution can be treated as an initial-value problem. That is, the instantaneous variation with time of the probability distribution of possible states of the process is determined by the current distribution, which includes the effects of all past history of the process.

An important figure in probability theory and the theory of random processes in the twentieth century was the Russian academician Andrei Nikolaeovich Kolmogorov (1903–1987). Starting around 1925, working with Aleksandr Yakovlevich Khinchin and others, he reestablished the foundations of probability theory on *measure theory*, which had originated as the basis for integration theory and became the accepted mathematical basis of probability and random processes. Along with Norbert Wiener, he is credited with founding much of the theory of prediction, smoothing and filtering of Markov processes, and the general theory of ergodic processes. His was the first formal theory of optimal estimation for systems involving random processes.

1.2.5 The Wiener–Kolmogorov Filter

Norbert Wiener (1894–1964) is one of the more famous prodigies of the early twentieth century. He was taught by his father until the age of 9, when he entered high

school. He finished high school at the age of 11 and completed his undergraduate degree in mathematics in 3 years at Tufts University. He then entered graduate school at Harvard University at the age of 14 and completed his doctorate degree in the philosophy of mathematics when he was 18. He studied abroad and tried his hand at several jobs for 6 more years. Then, in 1919, he obtained a teaching appointment at the Massachusetts Institute of Technology (MIT). He remained on the faculty of MIT for the rest of his life.

In the popular scientific press, Wiener is probably more famous for naming and promoting *cybernetics* than for developing the Wiener–Kolmogorov filter. Some of his greatest mathematical achievements were in generalized harmonic analysis, in which he extended the Fourier transform to functions of finite *power*. Previous results were restricted to functions of finite *energy*, which is an unreasonable constraint for signals on the real line. Another of his many achievements involving the generalized Fourier transform was proving that the transform of white noise is also white noise.[5]

1.2.5.1 Wiener–Kolmogorov Filter Development In the early years of World War II, Wiener was involved in a military project to design an automatic controller for directing antiaircraft fire with radar information. Because the speed of the airplane is a nonnegligible fraction of the speed of bullets, this system was required to "shoot into the future." That is, the controller had to predict the future course of its target using noisy radar tracking data.

In his derivation of an optimal estimator, Wiener would use probability measures on function spaces to represent uncertain dynamics. He derived the solution for the least-mean-squared prediction error in terms of the autocorrelation functions of the signal and the noise. The solution is in the form of an integral operator that can be synthesized with analog circuits, given certain constraints on the regularity of the autocorrelation functions or, equivalently, their Fourier transforms. His approach represents the probabilistic nature of random phenomena in terms of power spectral densities.

An analogous derivation of the optimal linear predictor for discrete-time systems was published by A. N. Kolmogorov in 1941, when Wiener was just completing his work on the continuous-time predictor.

Wiener's work was not declassified until the late 1940s, in a report titled "Extrapolation, Interpolation, and Smoothing of Stationary Time Series." The title was subsequently shortened to "Time Series." An early edition of the report had a yellow cover, and it came to be called the "yellow peril." It was loaded with mathematical details beyond the grasp of most engineering undergraduates, but it was absorbed and used by a generation of dedicated graduate students in electrical engineering.

[5]He is also credited with the discovery that the power spectral density of a signal equals the Fourier transform of its autocovariance function, although it was later discovered that Albert Einstein had known it before him.

1.2.6 The Kalman Filter

Rudolf Emil Kalman was born on May 19, 1930, in Budapest, the son of Otto and Ursula Kalman. The family emigrated from Hungary to the United States during World War II. In 1943, when the war in the Mediterranean was essentially over, they traveled through Turkey and Africa on an exodus that eventually brought them to Youngstown, Ohio, in 1944. Rudolf attended Youngstown College there for 3 years before entering MIT.

Kalman received his bachelor's and master's degrees in electrical engineering at MIT in 1953 and 1954, respectively. His graduate advisor was Ernst Adolph Guillemin, and his thesis topic was the behavior of solutions of second-order difference equations [142]. When he undertook the investigation, it was suspected that second-order difference equations might be modeled by something analogous to the describing functions used for second-order differential equations. Kalman discovered that their solutions were not at all like the solutions of differential equations. In fact, they were found to exhibit chaotic behavior.

In the fall of 1955, after a year building a large analog control system for the E. I. Du Pont Company, Kalman obtained an appointment as lecturer and graduate student at Columbia University. At that time, Columbia was well known for the work in control theory by John R. Ragazzini, Lotfi A. Zadeh,[6] and others. Kalman taught at Columbia until he completed the doctor of science degree there in 1957.

For the next year, Kalman worked at the research laboratory of the International Business Machines Corporation in Poughkeepsie and for 6 years after that at the research center of the Glenn L. Martin company in Baltimore, the Research Institute for Advanced Studies (RIAS).

To head its mathematics division, RIAS had lured the mathematician Solomon Lefschetz (1884–1972) from Princeton University. Lefschetz had been a classmate of the rocket pioneer Robert H. Goddard (1882–1945) at Clark University and thesis advisor to Richard E. Bellman (1920–1984) at Princeton. Lefschetz hired Kalman on the recommendation of Robert W. Bass, who had been a postdoctoral student under Lefschetz at Princeton before coming to RIAS in 1956. Kalman recommended Richard S. Bucy, who would join him at RIAS.

1.2.6.1 Discovery of the Kalman Filter In 1958, the Air Force Office of Scientific Research (AFOSR) was funding Kalman and Bucy to do advanced research in estimation and control at RIAS.

In late November 1958, not long after coming to RIAS, Kalman was returning by train to Baltimore from a visit to Princeton. At around 11 P.M., the train was halted for about an hour just outside Baltimore. It was late, he was tired, and he had a headache. While he was trapped there on the train for that hour, an idea occurred to him: *Why not apply the notion of state variables*[7] *to the Wiener–Kolmogorov filtering*

[6]Zadeh is perhaps more famous as the father of fuzzy systems theory and interpolative reasoning.

[7]Although frequency-domain methods were then the preferred approach to the filtering problem, the use of time-domain state-space models for time-varying systems had already been introduced (e.g., by Laning and Battin [163] in 1956).

problem? He was too tired to think much more about it that evening, but it marked the beginning of a great exercise to do just that. He read through Loève's book on probability theory [177] and equated expectation with projection. That proved to be pivotal in the derivation of the Kalman filter.

The Kalman filter was the culmination of a progression of models and associated optimal estimation methods for dynamic processes.

1. Wiener–Kolmogorov models use the power spectral density (PSD) in the frequency domain to characterize the dynamic and statistical properties of a dynamic process. Optimal Wiener–Kologorov estimators are derivable from the PSD, which can be estimated from measured system outputs. This assumes that the dynamic process model is time-invariant.
2. Control theorists use linear differential equations as dynamic system models. This led to the development of mixed models, in which the dynamic system functions as a "shaping filter" excited by white noise. Coefficients of the linear differential equations determine the shape of the output PSD, and the shape of the PSD defines the Wiener–Kolmogorov estimator. This approach allows the dynamic system model to be time-varying. These linear differential equations can be modeled as a system of first-order differential equations in what has come to be called *state space*.

The next step in this progression would be to develop the equivalent estimation methods right from a time-varying state-space model—and that is what Kalman did.

According to Kalman and Robert W. Bass [22], who was at RIAS in that period, it was Richard S. Bucy who recognized that—if one assumes a finite-dimensional state-space model—the Wiener–Hopf equation used in deriving the Wiener–Kolmogorov filter is equivalent to a nonlinear matrix-valued differential equation. Bucy also recognized that the nonlinear differential equation in question was of the same type as one studied by Jacopo Francesco Riccati (1676–1754) more than two centuries earlier, now called *the Riccati equation*. The general nature of this relationship between integral equations and differential equations first became apparent around that time. One of the more remarkable achievements of Kalman and Bucy in that period was proving that the Riccati equation can have a stable (steady-state) solution even if the dynamic system is unstable—provided that the system is observable and controllable.

With the additional assumption of finite dimensionality, Kalman was able to derive the Wiener–Kolmogorov filter as what we now call the Kalman filter. With the change to state-space form, the mathematical background needed for the derivation became much simpler, and the proofs were within the mathematical reach of many undergraduates.

Earlier Results The Danish astronomer Thorvald Nicolai Thiele (1838–1910) had derived what is essentially the Kalman filter for scalar processes, and some of

the seminal ideas in the Kalman filter had been published by Peter Swerling (1929–2001) in 1959 [265] and Ruslan Leont'evich Stratonovich (1930–1997) in 1960 [133, 261]:

1.2.6.2 Introduction of the Kalman Filter Kalman's ideas were met with some skepticism among his peers, and he chose a mechanical engineering journal (rather than an electrical engineering journal) for publication, because "When you fear stepping on hallowed ground with entrenched interests, it is best to go sideways."[8] His second paper, on the continuous-time case and coauthored with Bucy, was once rejected because—as one referee put it—one step in the proof "cannot possibly be true." (It was true.) He persisted in presenting his filter, and there was more immediate acceptance elsewhere. It soon became the basis for research topics at many universities and the subject of hundreds of doctoral theses in electrical engineering over the next decade or so.

1.2.6.3 Early Applications: The Influence of Stanley F. Schmidt Kalman found a receptive audience for his filter in the fall of 1960 in a visit to Stanley F. Schmidt at the Ames Research Center of NASA in Mountain View, California [186]. Schmidt had know Kalman from meetings at technical conferences and had invited him to Ames to further explain his approach. Schmidt had recognized its potential applicability to a problem then being studied at Ames—the trajectory estimation and control problem for the Apollo project, a planned manned mission to the moon and back. Schmidt began work immediately on what was probably the first full implementation of the Kalman filter. He soon discovered what is now called *extended Kalman filtering*, which has been used ever since for many real-time nonlinear applications of Kalman filtering. Enthused over his own success with the Kalman filter, he set about proselytizing others involved in similar work. In early 1961, Schmidt described his results to Richard H. Battin of the MIT Instrumentation Laboratory (later renamed the Charles Stark Draper Laboratory). Battin was already using state-space methods for the design and implementation of astronautical guidance systems, and he made the Kalman filter part of the Apollo onboard guidance system, which was designed and developed at the Instrumentation Laboratory. In the mid-1960s, through the influence of Schmidt, the Kalman filter became part of the Northrup-built navigation system for the C5A air transport, then being designed by Lockheed Aircraft Company. The Kalman filter solved the *data fusion problem* associated with combining radar data with inertial sensor data to arrive at an overall estimate of the aircraft trajectory and the *data rejection problem* associated with detecting exogenous errors in measurement data. It has been an integral part of nearly every onboard trajectory estimation and control system designed since that time.

[8]The two quoted segments in this paragraph are from a talk on "System Theory: Past and Present" given by Kalman at the University of California at Los Angeles (UCLA) on April 17, 1991, in a symposium organized and hosted by A. V. Balakrishnan at UCLA and sponsored jointly by UCLA and the National Aeronautics and Space Administration's (NASA) Dryden Laboratory.

1.2.6.4 Other Accomplishments of R. E. Kalman Around 1960, Kalman showed that the related notion of observability for dynamic systems had an algebraic dual relationship with controllability. That is, by the proper exchange of system parameters, one problem could be transformed into the other, and vice versa.

Kalman also played a leading role in the development of *realization theory*, which also began to take shape around 1962. This theory addresses the problem of finding a system model to explain the observed input/output behavior of a system. This line of investigation led to a *uniqueness principle* for the mapping of exact (i.e., noiseless) data to linear system models.

In 1985, Kalman was awarded the Kyoto Prize, considered by some to be the Japanese equivalent of the Nobel Prize. On his visit to Japan to accept the prize, he related to the press an epigram that he had first seen in a pub in Colorado Springs in 1962, and it had made an impression on him. It said:

> Little people discuss other people.
>
> Average people discuss events.
>
> Big people discuss ideas.[9]

His own work, he felt, had been concerned with ideas.

In 1990, on the occasion of Kalman's sixtieth birthday, a special international symposium was convened for the purpose of honoring his pioneering achievements in what has come to be called *mathematical system theory*, and a *Festschrift* with that title was published soon after [15].

1.2.6.5 Impact of Kalman Filtering on Technology From the standpoint of those involved in estimation and control problems, at least, this has to be considered the greatest achievement in estimation theory of the twentieth century. Many of the achievements since its introduction would not have been possible without it. It was one of the enabling technologies for the Space Age, in particular. The precise and efficient navigation of spacecraft through the solar system could not have been done without it.

The principal uses of Kalman filtering have been in modern control systems, in the tracking and navigation of all sorts of vehicles, and in predictive design of estimation and control systems. These technical activities were made possible by the introduction of the Kalman filter. (If you need a demonstration of its impact on technology, enter the keyword "Kalman filter" in a technical literature search. You will be overwhelmed by the sheer number of references it will generate.)

1.2.6.6 Relative Advantages of Kalman and Wiener–Kolmogorov Filtering

1. The Wiener–Kolmogorov filter implementation in analog electronics can operate at much higher effective throughput than the (digital) Kalman filter.

[9]A different phrasing of this epigram has been attributed to columnist Ann Landers (Esther Pauline Friedman Lederer, 1918–2002).

2. The Kalman filter is implementable in the form of an algorithm for a digital computer, which was replacing analog circuitry for estimation and control at the time that the Kalman filter was introduced. This implementation may be slower, but it is capable of much greater accuracy than had been achievable with analog filters.
3. The Wiener–Kolmogorov filter does not require finite-dimensional stochastic process models for the signal and noise.
4. The Kalman filter does not require that the deterministic dynamics or the random processes have stationary properties, and many applications of importance include nonstationary stochastic processes.
5. The Kalman filter is compatible with the state-space formulation of optimal controllers for dynamic systems, and Kalman was able to prove useful dual properties of estimation and control for these systems.
6. For the modern controls engineering student, the Kalman filter requires less additional mathematical preparation to learn and use than the Wiener–Kolmogorov filter. As a result, the Kalman filter can be taught at the undergraduate level in engineering curricula.
7. The Kalman filter provides the necessary information for mathematically sound, statistically based decision methods for detecting and rejecting anomalous measurements.

1.2.7 Implementation Methods

1.2.7.1 Numerical Stability Problems The great success of Kalman filtering required the solution of certain implementation problems, not the least of which was the marginal stability of the numerical solution of the associated Riccati equation. In some applications, small roundoff errors tended to accumulate and eventually degrade the performance of the filter. In the decades immediately following the introduction of the Kalman filter, there appeared several better numerical implementations of the original formulas. Many of these were adaptations of methods previously derived for the least-squares problem.

1.2.7.2 Early Ad Hoc Fixes It was discovered early on[10] that forcing symmetry on the solution of the matrix Riccati equation improved its apparent numerical stability—a phenomenon that was later given a more theoretical basis by Verhaegen and Van Dooren [278]. It was also found that the influence of roundoff errors could be ameliorated by artificially increasing the covariance of process noise in the Riccati equation. A symmetrized form of the discrete-time Riccati equation was developed by Joseph [56] and used by R. C. K. Lee at Honeywell in 1964. This "structural" reformulation of the Kalman filter equations improved robustness against roundoff

[10]These fixes were apparently discovered independently by several people. Schmidt [186] and his colleagues at NASA had discovered the use of forced symmetry and "pseudonoise" to counter roundoff effects and cite R. C. K. Lee at Honeywell with the independent discovery of the symmetry effect.

errors in some applications, although later methods have performed better on some problems [269].

1.2.7.3 James E. Potter (1937–2005) and Square-Root Filtering The biggest breakthrough in improving the numerical stability of Kalman filtering occurred at the Instrumentation Laboratory at MIT, the prime contractor for guidance and control of the Apollo moon project. The Kalman filter for Apollo navigation could be implemented in 36-bit floating-pointing arithmetic on an IBM 700-series mainframe computer, but it would eventually have to run on a flight computer using 15-bit fixed-point arithmetic. The main problem was implementing the Riccati equation solution. James Potter was then an MIT graduate student working part-time at the laboratory. He took the problem home with him on a Friday and came back on the following Monday with the solution.

Potter introduced the idea of factoring the covariance matrix into *Cholesky factors*,[11] in the format

$$P = CC^{\mathrm{T}}, \tag{1.14}$$

and expressing the observational update equations in terms of the Cholesky factor C, rather than P. The result was better numerical stability of the filter implementation at the expense of added computational complexity. A generalization of the Potter method to handle vector-valued measurements was published by one of the authors [10] in 1968, but a more efficient implementation—in terms of *triangular* Cholesky factors—was published by Bennet in 1967 [29].

1.2.7.4 Improved Square-Root and UD Filters There was a rather rapid development of faster algorithmic methods for square-root filtering in the 1970s, following the work at NASA/JPL (then called the Jet Propulsion Laboratory, at the California Institute of Technology) in the late 1960s by Dyer and McReynolds [78] on temporal update methods for Cholesky factors. Extensions of square-root covariance and information filters were introduced in Kaminski's 1971 thesis [147] at Stanford University. The first of the triangular factoring algorithms for the observational update was due to Agee and Turner [2] in a 1972 report of rather limited circulation. These algorithms have roughly the same computational complexity as the conventional Kalman filter, but with better numerical stability. The *fast triangular* algorithm of Carlson was published in 1973 [57], followed by the "square-root-free" algorithm of Bierman in 1974 [31] and the associated temporal update method introduced by Thornton [268]. The computational complexity of the square-root filter for time-invariant systems was greatly simplified by Morf and Kailath [201] soon afterward.

[11] A square root S of a matrix P satisfies the equation $P=SS$ (i.e., without the transpose on the second factor). Potter's derivation used a special type of symmetric matrix called an *elementary matrix*. He factored an elementary matrix as a square of another elementary matrix. In this case, the factors were truly square roots of the factored matrix. This square-root appellation has stuck with extensions of Potter's approach, even though the factors involved are Cholesky factors, not matrix square roots.

Specialized parallel processing architectures for fast solution of the square-root filter equations were developed by Jover and Kailath [119] and others over the next decade, and much simpler derivations of these and earlier square-root implementations were discovered by Kailath [139].

1.2.7.5 Factorization Methods The square-root methods make use of matrix decomposition[12] methods that were originally derived for the least-squares problem. These include the so-called *QR* decomposition of a matrix as the product of an orthogonal matrix (Q) and a "triangular"[13] matrix (R). The matrix R results from the application of orthogonal transformations of the original matrix. These orthogonal transformations tend to be well conditioned numerically. The operation of applying these transformations is called the *triangularization* of the original matrix, and triangularization methods derived by Givens [97], Householder [112], and Gentleman [96] are used to make Kalman filtering more robust against roundoff errors.

1.2.8 Adaptations for Nonlinear Filtering

As the psychologist Abraham Maslow (1908–1970) once observed:

> When the only tool you have is a hammer, every problem begins to resemble a nail.

The same tendency applies when that tool is a Kalman filter. Those experienced with Kalman filtering often find themselves morphing problems to resemble the Kalman filtering model.

This is especially so with nonlinear problems, for which there is no practical mathematical approach comparable to the Kalman filter. Although it was originally derived for linear problems, the Kalman filter is habitually applied to nonlinear problems. This approach has worked remarkably well for a number of nonlinear problems, but there will always be limits to how far it can be pushed.

We mention here some approaches that have been used to extend the applicability of Kalman filtering methodologies to nonlinearly problems. The more successful of these are described in greater detail in Chapter 6.

1.2.8.1 Extended Kalman Filtering for Quasilinear Problems Extended Kalman filtering was used in the very first application of Kalman filtering: the space navigation problem for the Apollo missions to the moon and back. The approach has been successfully applied to many nonlinear problems ever since.

Success depends on the problem's being *quasilinear*, in the sense that unmodeled errors due to linear approximation are insignificant compared to the modeled errors

[12]The term *decomposition* refers to the representation of a matrix (in this case, a covariance matrix) as a product of matrices having more useful computational properties, such as sparseness (for triangular factors) or good numerical stability (for orthogonal factors). The term *factorization* was used by Bierman [31] for such representations.

[13]See Chapter 6 and Appendix B for discussions of triangular forms.

due to dynamic uncertainty and sensor noise. Methods for verifying whether a problem is quasi-linear are presented in Chapter 6.

In extended Kalman filtering, linear approximation is used only for solving the Riccati equation, a partial result of which is the Kalman gain. The full nonlinear model is used in propagation of the estimate and in computing predicted sensor outputs.

The approach uses partial derivatives as linear approximations of nonlinear relations. Schmidt [186] introduced the idea of evaluating these partial derivatives at the *estimated* value of the state variables. This and other methods for approximate linear solutions to nonlinear problems are discussed in Chapter 6.

1.2.8.2 Higher Order Approximations Approaches using higher order expansions of the filter equations (i.e., beyond the linear terms) have been derived by Stratonovich [262], Kushner [158], Bucy [54], Bass et al. [23], and others for quadratic nonlinearities and by Wiberg and Campbell [282] for terms through the third order.

1.2.8.3 Sampling-Based Methods for Nonlinear Estimation Extended Kalman filtering is designed for quasilinear problems, in which the unmodeled errors introduced by linear approximation in the Riccati equation are insignificant compared to the modeled errors from measurement noise and dynamic uncertainty.

The Kalman filtering methodology has been further extended to nonquasilinear problems by using nonlinear propagation of the Riccati equation solution by representative samples of state and measurement variables—as opposed to linearized propagation of the mean (i.e., the estimated state) and covariance matrix of the distribution.

In the 1940s, Nicholas Metropolis used the term *Monte Carlo* to describe analytical methods using the propagation of pseudorandom samples to represent changes in probability distributions brought about by transformations of the variate. (The name refers to the Monte Carlo gambling casino, which uses pseudorandom methods to transform the distribution of wealth among its players.)

Much of the initial development of Monte Carlo techniques occurred at the Los Alamos Laboratory, where adequate computer resources were then becoming available. Others involved in this development at Los Alamos included Stanislaw Ulam, Enrico Fermi, and John von Neumann.

The computational burden of sample-based analysis can be reduced significantly by using more judicious sampling rules, in place of random sampling:

1. In *sequential Monte Carlo* methods, the samples are selected in the order of their relative importance for representing the significant features of the distribution.
2. In *sigma-point* filtering, the samples are related to the means and standard deviations (usually represented by the symbol σ) of the distributions.

3. *Unscented transform* methods select samples based on the mean of the distribution and the eigenstructure of the covariance matrix. The resulting filter implementation is called *unscented Kalman filtering*, a term introduced by Simon Julier and Jeffrey Uhlmann. Unscented transformations for n-dimensional distributions generally require at least $2n + 1$ samples, which is about minimal for sample-based methods.

The term *particle filter* is also used to denote extensions of the Kalman filter that use sample-based methods, because the sampled values can be viewed as "particles" carried along by the nonlinear system dynamics.

In all cases, the samples of state vector values are chosen to represent the mean and covariance structure of the ensemble *a posteriori* distribution (i.e., after the measurement information has been used for refining the estimate). These sample points are then propagated forward in time by simulating the known nonlinear system dynamics, and the resulting *a priori* covariance at the next measurement opportunity is inferred from the resulting distribution after the nonlinear transformations of individual samples. The resulting covariance structure is then used in computing the Kalman gains to use the measured sensor outputs.

These methods have been shown to be efficient and effective for some nonquasi-linear applications—including system identification (i.e., estimation of dynamic model parameters), a notoriously nonlinear and difficult problem. The more successful of these methods are described in Chapter 6.

1.2.9 Truly Nonlinear Estimation

Problems involving nonlinear and random dynamic systems have been studied for some time in statistical mechanics. The propagation over time of the *probability distribution* of the state of a nonlinear dynamic system is described by a nonlinear partial differential equation called the *Fokker–Planck equation*. It has been studied by Einstein [80], Fokker [85], Planck [214], Kolmogorov [154], Stratonovich [262], Baras and Mirelli [21], and others. Stratonovich modeled the effect on the probability distribution of information obtained through noisy measurements of the dynamic system, an effect he called *conditioning*. The partial differential equation that includes these effects is called the *conditioned Fokker–Planck equation*. It has also been studied by Kushner [158], Bucy [54], and others using the *stochastic calculus* of Stratonovich or Kiyosi Itô. The theoretical basis for stochastic differential equations was long hampered by the fact that white noise is not a Riemann-integrable function, but the non-Riemannian *stochastic integrals* of Stratonovich or Itô solved that problem.

The general approach results in a stochastic partial differential equation describing the evolution over time of the probability distribution over a state space of the dynamic system under study. The resulting models do *not* enjoy the finite representational characteristics of the Kalman filter, however. The computational complexity of obtaining a solution far exceeds the already considerable burden of the

conventional Kalman filter. These methods are of significant interest and utility but are beyond the scope of this book.

1.2.10 The Detection Problem for Surveillance

Surveillance problems include the detection, identification, and tracking of objects within a certain region of space. The Kalman filter helps solve the tracking problem and may be of some utility (as a nonlinear filter) in solving the identification problem. However, the *detection problem* must usually be solved before identification and tracking can begin. The Kalman filter requires an initial state estimate for each object, and that initial estimate must be obtained by detecting it. Those initial states are distributed according to some point process, but there are no technically mature methods (comparable to the Kalman filter) for estimating the state of a point process.

A *point process* is a random process for modeling events or objects that are distributed over time and/or space, such as the arrivals[14] of messages at a communications switching center or the locations of stars in the sky. It is also a model for the initial states of systems in many estimation problems, such as the locations in time and space of aircraft or spacecraft under surveillance by a radar installation, or the locations of submarines under sonar surveillance in the ocean.

A unified approach combining detection and tracking into one optimal estimation method was derived by John M. Richardson (1918–1996) and specialized to several applications [232]. The detection and tracking problem for a *single object* is represented by the conditioned Fokker–Planck equation. Richardson derived from this one-object model an infinite hierarchy of partial differential equations representing *object densities* and truncated this hierarchy with a simple closure assumption about the relationships between orders of densities. The result is a single partial differential equation approximating the evolution of the density of objects. It can be solved numerically. It provides a solution to the difficult problem of detecting dynamic objects whose initial states are represented by a point process.

1.3 ON THE NOTATION USED IN THIS BOOK

1.3.1 Symbolic Notation

The fundamental problem of symbolic notation, in almost any context, is that there are never enough symbols to go around. There are not enough letters in the Roman alphabet to represent the basic phonetic elements of standard spoken English, let alone all the variables in Kalman filtering and its applications. As a result, some symbols must play multiple roles. In such cases, their roles will be defined as they are introduced. This is sometimes confusing, but it is unavoidable.

[14]In these applications, a point process is also called an *arrival process*.

TABLE 1.2 Standard Symbols of Kalman Filtering

Symbols			
I*	II†	III‡	Symbol Definition
F	F	A	Dynamic coefficient matrix of a continuous linear differential equation defining a dynamic system
G	I	B	Coupling matrix between random process noise and the state of a linear dynamic system
H	M	C	Measurement sensitivity matrix defining the linear relationship between the state of the dynamic system and measurements that can be made
$\overline{K}$	Δ	K	Kalman gain matrix
P	P		Covariance matrix of state estimation uncertainty
Q	Q		Covariance matrix of process noise in the system state dynamics
R	0		Covariance matrix of observational (measurement) uncertainty
x	x		State vector of a linear dynamic system
z	y		Vector (or scalar) of measured values
Φ	Φ		State transition matrix of a discrete linear dynamic system

*This book [8, 51, 58, 95].
†Kalman [143].
‡Other sources [20, 46, 64, 134].

1.3.1.1 "Dot" Notation for Derivatives Newton's notation using $\dot{f}(t), \ddot{f}(t)$ for the first two derivatives of f with respect to t is used where convenient to save ink.

1.3.1.2 Standard Symbols for Kalman Filter Variables There appear to be two standard conventions in technical publications for the symbols used in Kalman filtering. The one used in this book is similar to the original notation of Kalman [143]. The other standard notation is sometimes associated with applications of Kalman filtering in control theory. It uses the first few letters of the alphabet in place of the Kalman notation. Both sets of symbol usages are presented in Table 1.2, along with the original (Kalman) notation.

1.3.1.3 State Vector Notation for Kalman Filtering The state vector x has been adorned with all sorts of other appendages in the usage of Kalman filtering. Table 1.3 lists the notation used in this book (left column) along with notations found in some other sources (second column). The state vector wears a "hat" as the estimated value, $\hat{x}$, and subscripting to denote the sequence of values that the estimate assumes over time. The problem is that it has two values at the same time: the *a priori*[15] value (before the measurement at the current time has been used in refining the estimate) and the *a posteriori* value (after the current measurement has been used in refining

[15]This use of the full Latin phrases as adjectives for the prior and posterior statistics is an unfortunate choice of standard notation, because there is no easy way to shorten it. (Even their initial abbreviations are the same.) If those who initiated this notation had known how commonplace the notation would become, they might have named the phrases otherwise.

TABLE 1.3 Special State-Space Notation

This Book	Other Sources	Definition of Notational Usage
x	x, $\bar{x}$, x	State vector
x_k		The kth component of the vector x
x_k	$x[k]$	The kth element of the sequence $\ldots, x_{k-1}, x_k, x_{k+1}, \ldots$ of vectors
$\hat{x}$	$\mathrm{E}\langle x\rangle$, $\bar{x}$	An estimate of the value of x
$\hat{x}_k(-)$	$\hat{x}_{k\|k-1}$, $\hat{x}_{k-}$	*A priori* estimate of x_k, conditioned on all prior measurements except the one at time t_k
$\hat{x}_k(+)$	$\hat{x}_{k\|k}$, $\hat{x}_{k+}$	*A posteriori* estimate of x, conditioned on all available measurements at time t_k
$\dot{x}$	x_t, dx/dt	Derivative of x with respect to t (time)

TABLE 1.4 Common Notation for Array Dimensions

					Dimensions	
Symbol	Vector Name	Dimensions	Symbol	Matrix Name	Row	Column
x	System state	n	Φ	State transition	n	n
w	Process noise	r	G	Process noise coupling	n	r
u	Control input	r	Q	Process noise covariance	r	r
z	Measurement	ℓ	H	Measurement sensitivity	ℓ	n
v	Measurement noise	ℓ	R	Measurement noise covariance	ℓ	ℓ

the estimate). These distinctions are indicated by the signum. The negative sign $(-)$ indicates the *a priori* value, and the positive sign $(+)$ indicates the *a posteriori* value.

1.3.1.4 Common Notation for Array Dimensions Symbols used for the *dimensions* of the standard arrays in Kalman filtering will also be standardized, using the notation of Gelb et al. [95] shown in Table 1.4. These symbols are not used exclusively for these purposes. (Otherwise, one would soon run out of alphabet.) However, whenever one of these arrays is used in the discussion, these symbols will be used for their dimensions.

1.4 SUMMARY

The *Kalman filter* is an estimator used to estimate the *state* of a *linear dynamic system* perturbed by *Gaussian white noise* using measurements that are *linear* functions of the system

state but corrupted by *additive* Gaussian white noise. The mathematical model used in the derivation of the Kalman filter is a reasonable representation for many problems of practical interest, including *control problems* as well as *estimation problems*. The Kalman filter model is also used for the *analysis* of *measurement and estimation problems*.

The *method of least squares* was the first optimal estimation method. It was discovered by Gauss (and others) around the end of the eighteenth century, and it is still much in use today. If the associated *Gramian matrix* is *nonsingular*, the method of least squares determines the *unique* values of a set of unknown variables such that the *squared deviation* from a set of constraining equations is minimized.

Observability of a set of unknown variables is the issue of whether or not they are *uniquely determinable* from a given set of *constraining equations*. If the constraints are *linear* functions of the unknown variables, then those variables are *observable if and only if* the associated Gramian matrix is nonsingular. If the Gramian matrix is *singular*, then the unknown variables are *unobservable*.

The *Wiener–Kolmogorov filter* was derived in the 1940s by Norbert Wiener (using a model in continuous time) and Andrei Kolmogorov (using a model in discrete time) working independently. It is a *statistical estimation method*. It estimates the state of a dynamic process so as to minimize the *mean-squared estimation error*. It can take advantage of *statistical* knowledge about random processes in terms of their PSDs in the *frequency domain*.

The *state-space model* of a dynamic process uses differential equations (or difference equations) to represent both deterministic and random phenomena. The *state variables* of this model are the variables of interest and their derivatives of interest. Random processes are characterized in terms of their statistical properties in the *time domain* rather than the frequency domain. The Kalman filter was derived as the solution to the Wiener filtering problem using the state-space model for dynamic and random processes. The result is easier to derive (and to use) than the Wiener–Kolmogorov filter.

Square-root filtering is a reformulation of the Kalman filter for better numerical stability in finite-precision arithmetic. It is based on the same mathematical model, but it uses an equivalent statistical parameter that is less sensitive to roundoff errors in the computation of optimal filter gains. It incorporates many of the more numerically stable computation methods that were originally derived for solving the least-squares problem.

Sequential Monte Carlo methods can be used to extend Kalman filtering beyond the *quasilinear* estimation problems that are solvable by *extended Kalman filtering*.

PROBLEMS

1.1 Derive the least-squares equations for finding a and b which provide the best fit to the three equations

$$\left.\begin{aligned} 2 &= a + b \\ 1 &= 2a + b \\ 4 &= 3a + b \end{aligned}\right\} \tag{1.15}$$

(**a**) Express the system of Equations in 1.15 in matrix form as

$$z = A\begin{bmatrix} a \\ b \end{bmatrix},$$

where z is a column vector with three rows and the matrix A is 3×2.

(**b**) Take the matrix product $A^{\mathrm{T}}A$ for A derived in (a).

(**c**) Take the 2×2 matrix inverse

$$[A^{\mathrm{T}}A]^{-1}$$

for the A derived in (a). [Hint: Use the general formula

$$\begin{bmatrix} m_{11} & m_{12} \\ m_{12} & m_{22} \end{bmatrix}^{-1} = \frac{1}{m_{11}m_{22} - m_{12}^2}\begin{bmatrix} m_{22} & -m_{12} \\ -m_{12} & m_{11} \end{bmatrix} \tag{1.16}$$

for inverting symmetric 2×2 matrices.]

(**d**) Take the matrix product

$$A^{\mathrm{T}}\mathcal{Z}$$

for the A derived in (a).

(**e**) Calculate the least-squares solution

$$\begin{bmatrix} \hat{a} \\ \hat{b} \end{bmatrix} = [A^{\mathrm{T}}A]^{-1}\, A^{\mathrm{T}}\mathcal{Z}$$

for $[A^{\mathrm{T}}A]^{-1}$ derived in (c) and $A^{\mathrm{T}}\mathcal{Z}$ derived in (d).

1.2 Find the least-squares solution for a and b in the four equations

$$\begin{aligned} 2 &= a + b \\ 1 &= 2a + b \\ 4 &= 3a + b \\ 4 &= 4a + b. \end{aligned}$$

1.3 The straight-line-fit problem with uniform sampling is to find a bias b and ramp coefficient a to fit a set of N measured values $z_1, z_2, z_3, \ldots, z_N$ sampled at

uniform time intervals Δt. The problem can modeled by a system of N linear equations

$$\left.\begin{array}{rcl} z_1 &=& a1\Delta t + b \\ z_2 &=& a2\Delta t + b \\ z_3 &=& a3\Delta t + b \\ \vdots & \vdots & \vdots \\ z_N &=& aN\Delta t + b, \end{array}\right\} \tag{1.17}$$

where the unknowns are a and b.

(**a**) Express the system of equations in matrix form, using dots to represent A in the form

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \\ \vdots & \vdots \\ a_{N1} & a_{N2} \end{bmatrix},$$

but with formulas for the matrix elements a_{ij}.

(**b**) Derive a symbolic formula for the 2×2 matrix $A^{\mathrm{T}}A$ for A as defined in (a).

(**c**) Use Equation 1.16 to derive a formula for $[A^{\mathrm{T}}A]^{-1}$ for the $A^{\mathrm{T}}A$ defined in (b).

(**d**) Use the above results to derive formulas for the least-squares estimates $\hat{a}$, $\hat{b}$ for the general system of N linear equations in Equation 1.17.

1.4 Jean Baptiste Fourier (1768–1830) was studying the problem of approximating a function $f(\theta)$ on the circle $0 \leq \theta < 2\pi$ by a linear combination of trigonometric functions:

$$f(\theta) \approx a_0 + \sum_{j=1}^{n} [a_j \cos(j\theta) + b_j \sin(j\theta)]. \tag{1.18}$$

See if you can help him solve this problem. Use the method of least squares to demonstrate that the values

$$\hat{a}_0 = \frac{1}{2\pi} \int_0^{2\pi} f(\theta)\, d\theta,$$

$$\hat{a}_j = \frac{1}{\pi} \int_0^{2\pi} f(\theta) \cos(j\theta)\, d\theta,$$

$$\hat{b}_j = \frac{1}{\pi} \int_0^{2\pi} f(\theta) \sin(j\theta)\, d\theta$$

of the coefficients a_j and b_j for $1 \le j \le n$ give the least integrated squared approximation error

$$\begin{aligned}
\varepsilon^2(a,b) &= ||f - \hat{f}(a,b)||^2_{\mathcal{L}_2} \\
&= \int_0^{2\pi} [\,\hat{f}(\theta) - f(\theta)]^2 \, d\theta \\
&= \int_0^{2\pi} \left\{ a_0 + \sum_{j=1}^{n} [a_j \cos(j\theta) + b_j \sin(j\theta)] \right\}^2 d\theta \\
&\quad - 2 \int_0^{2\pi} \left\{ a_0 + \sum_{j=1}^{n} [a_j \cos(j\theta) + b_j \sin(j\theta)] \right\} f(\theta) \, d\theta \\
&\quad + \int_0^{2\pi} f^2(\theta) \, d\theta.
\end{aligned}$$

You may assume the equalities

$$\int_0^{2\pi} d\theta = 2\pi$$

$$\int_0^{2\pi} \cos(j\theta)\cos(k\theta)\, d\theta = \begin{cases} 0, & j \neq k \\ \pi, & j = k \end{cases}$$

$$\int_0^{2\pi} \sin(j\theta)\sin(k\theta)\, d\theta = \begin{cases} 0, & j \neq k \\ \pi, & j = k \end{cases}$$

$$\int_0^{2\pi} \cos(j\theta)\sin(k\theta)\, d\theta = 0, \quad 0 \le j \le n, \quad 1 \le k \le n$$

as given.

2

LINEAR DYNAMIC SYSTEMS

What we experience of nature is in models, and all of nature's models are so beautiful.[1]
R. Buckminster Fuller (1895–1983)

2.1 CHAPTER FOCUS

This chapter is about the dynamic models used in Kalman filtering, especially those represented by systems of linear differential equations.

The objective will be to demonstrate, using specific examples, how one goes about building such models, and how one can go from a model using differential equations to one suitable for Kalman filtering.

Where one gets these differential equations depends on the application. For modeling electromechanical systems, these differential equations generally come from the laws of physics.

For example, the differential equations for modeling many mechanical systems come from Newton's laws of mechanics,[2] such as $F = ma$, its rotational companion $T = M\dot{\omega}$, or Newton's laws of gravitation (used in Example 2.1).

[1] From an interview quoted by Calvin Tomkins in "From in the Outlaw Area," *The New Yorker*, January 8, 1966.
[2] F is the force vector acting on a rigid body with mass m, and a is the resulting acceleration vector of the body. T is the torque vector acting on the body, M is its moment matrix (i.e., the second moment of its mass distribution), and ω is the body's rotational rate vector.

Kalman Filtering: Theory and Practice using MATLAB®, Third Edition. By Mohinder S. Grewal and Angus P. Andrews

2.1.1 The Bigger Picture

How the dynamic models of this chapter fit into the overall estimation problem is illustrated in the simplified schematic of Figure 2.1, with variables as defined in Table 2.1. The schematic has been simplified to represent either continuous-time or discrete-time applications, the relationships between which are covered in this chapter.

The schematic shows three dynamic models:

- The one labeled "SYSTEM DYNAMICS" models the real-world dynamics in the intended application of the Kalman filter. This model may contain state variables for representing time-correlated random processes such as:
  - Time-correlated random dynamic disturbances, such as winds and (for ships) currents.
  - Time-correlated sensor noise, such as the effects of ambient temperature.

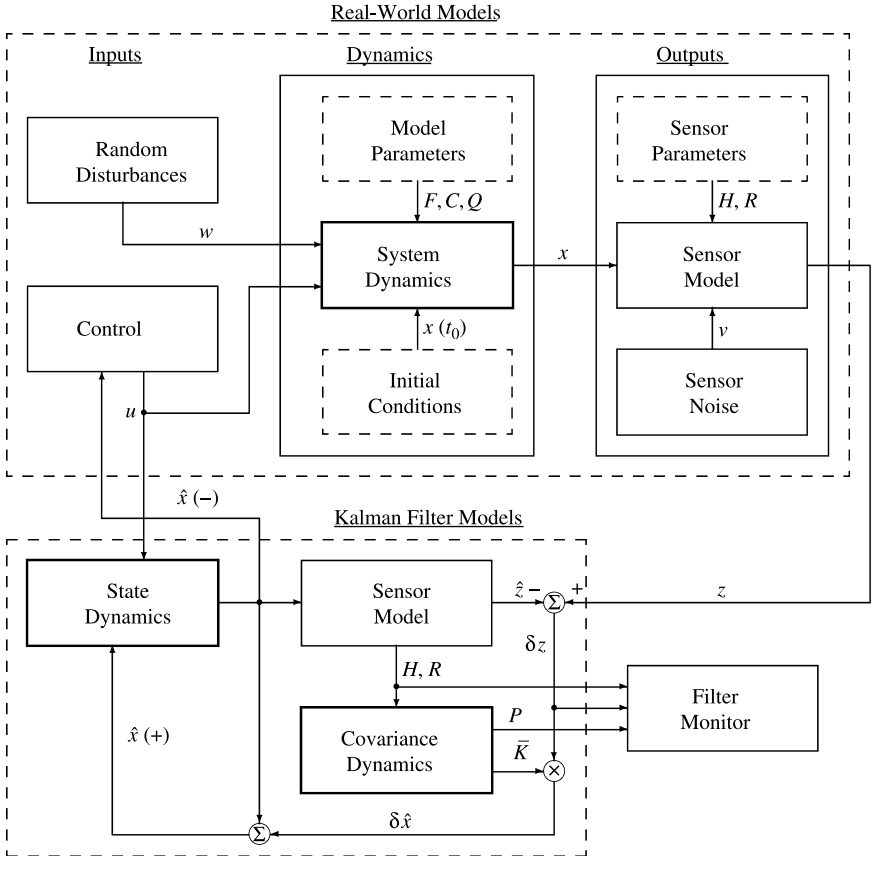


Figure 2.1 Schematic of Kalman filter implementation.

TABLE 2.1 Mathematical Models for Dynamic Systems

Model Type	Continuous Time Model	Discrete Time Model
Time invariant		
Linear	$\dot{x}(t) = Fx(t) + Cu(t) + w(t)$	$x_k = \Phi x_{k-1} + \Gamma u_{k-1} + w_{k-1}$
	$z(t) = Hx(t) + v(t)$	$z_k = Hx_k + v_k$
General	$\dot{x}(t) = f(x(t), u(t))$	$x_k = f(x_{k-1}, u_{k-1})$
	$z(t) = h(x(t), v(t))$	$z_k = h(x_k, v_k)$
Time varying		
Linear	$\dot{x}(t) = F(t)x(t) + C(t)u(t)$	$x_k = \Phi_{k-1}x_{k-1} + \Gamma_{k-1}u_{k-1}$
	$z(t) = H(t)x(t) + v(t)$	$z_k = H_k x_k + v_k$
General	$\dot{x}(t) = f(t, x(t), u(t))$	$x_k = f_k(x_{k-1}, u_{k-1})$
	$z(t) = h(t, x(t), v(t))$	$z_k = h_k(x_k, v_k)$
System inputs:		
u represents known control inputs.		
w represents random dynamic disturbances.		
v represents random sensor noise.		
System outputs:		
z represents sensor outputs.		

  - In some applications, the model may also contain as additional state variables the slowly varying "parameters" of its own dynamic model.

  Ideally, this model would be based on the laws of physics and the results of calibrating (with a Kalman filter) the real-world system dynamic parameters.

- The one labeled "STATE DYNAMICS" in the Kalman filter should contain a fairly faithful replication of the true system dynamics, except that it does not know the values of the random inputs w in the real-world model. The Kalman filter model may also include other variables to be estimated in addition to the true system state vector x. These additional variables might include estimates of parameters shown in the dashed boxes in Figure 2.1:
  - The initial conditions of the true system state vector $x(t_0)$. (The estimator used for this application is called a *fixed-point smoother*, which is described in Chapter 5.)
  - Matrix parameters F (or its discrete-time equivalent Φ) and Q (and possibly C) of the true system dynamic model, used as shown in Table 2.1. The estimator in this case becomes nonlinear, as discussed in Chapter 6.
  - Matrix parameters H and R of the sensor model, in which case the filter also becomes nonlinear. These sensor parameters may also include sensor output biases, for example.
  - The filter state variables may also include the means (biases) of the random disturbances w and/or sensor noise v. Additionally, the Kalman filter state variables may, in some applications, include the variances of the "RANDOM DISTURBANCES" (Q) and/or "SENSOR NOISE" (R) models, in which case the filter becomes nonlinear.

- The one labeled "COVARIANCE DYNAMICS" is a dynamic model for the second moment (covariance matrix P) of estimation errors. This is a *matrix Riccati equation* when the other models are linear or quasilinear (i.e., close enough to being linear that the errors due to nonlinearity do not matter). Otherwise, the propagation over time of the estimated mean and covariance of the state variable distribution may be implemented by using structured sampling methods (e.g., the unscented transform). Either way, the covariance update model includes the state dynamics model, and the implementation generates the Kalman gain matrix $\overline{k}$ as a partial result.

  The system input u (from CONTROL) is covered in this chapter, although no distinction is made between u and random dynamic disturbance w until Chapter 3. The even bigger picture (outside the scope of this book) includes the implementation inside the box labeled "CONTROL," which uses the estimated state $\hat{x}$ as input.

  The other parts of this higher-level schematic are covered in the following chapters.

- Random process models for the boxes labeled "RANDOM DISTURBANCES" and "SENSOR NOISE" are addressed in Chapter 3.
- Sensor models are addressed in Chapters 4 and 9.
- The Riccati equation is addressed in Chapter 4, and the equivalent methods for nonlinear applications are covered in Chapter 7.
- Methods for Kalman filter health monitoring (in the box labeled "FILTER MONITOR") are discussed in Chapter 8.

2.1.2 Models for Dynamic Systems

Differential Equations and State Variables Since their introduction by Newton and Leibniz in the seventeenth century, differential equations have provided concise and faithful mathematical models for many dynamic systems of importance to humans. By this device, Newton was able to model the motions of the planets in our solar system with a small number of variables and parameters. Given a finite number of initial conditions (accurate masses and initial positions and velocities of the sun and planets will do) and these equations, one can determine the positions and velocities of the planets relatively accurately for many years. The finite-dimensional representation of a problem (in this example, the problem of predicting the future course of the planets) is the basis for the so-called *state-space approach* to the representation of differential equations and their solutions, which is the focus of this chapter. The dependent variables of the differential equations become *state variables* of the dynamic system. They explicitly represent all the important characteristics of the dynamic system at any time.

Other Approaches Dynamic system theory is a subject of considerably more scope than one needs for the present undertaking (Kalman filtering). This chapter will focus on those concepts that are essential for that purpose, which is the development of the

state-space representation for dynamic systems described by systems of linear differential equations. These are given a somewhat heuristic treatment without the mathematical rigor often accorded the subject, omitting the development and use of the transform methods of functional analysis for solving differential equations when they serve no purpose in the derivation of the Kalman filter. The interested reader will find a more formal and thorough presentation in most upper-level and graduate-level textbooks on ordinary differential equations. The objective of the more engineering-oriented treatments of dynamic systems is usually to solve the *controls problem*, which is the problem of defining the *inputs* (i.e., control settings) that will bring the state of the dynamic system to a desirable condition. That is not the objective here, however.

2.1.3 Main Points to Be Covered

The objective in this chapter is to characterize the measurable *outputs*[3] of dynamic systems as functions of the internal *states* and *inputs* of the system. The treatment here is deterministic in order to define functional relationships between inputs and outputs.

In Chapter 3, the inputs are allowed to be nondeterministic (i.e., random), and the focus of Chapter 4 is on how to estimate the state variables of a dynamic system in this context.

Dynamic Systems and Differential Equations In the context of Kalman filtering, a *dynamic system* has come to be synonymous with a system of ordinary differential equations describing the evolution over time of the state of a physical system. This mathematical model is used to derive its solution, which specifies the functional dependence of the state variables on their initial values and the system inputs. This solution defines the functional dependence of the measurable outputs on the inputs and coefficients of the model.

Mathematical Models for Continuous and Discrete Time The principal dynamic system models of interest in this book are summarized in Table 2.1. These include linear models (the focus of this chapter) and nonlinear models (the focus in Chapter 7). They also include models in continuous time, as defined by differential equations, and models in discrete time, which are more suitable for computer implementation.

Modeling in Continuous Time How one goes about building linear differential equation models for real-world dynamic systems is the focus of Section 2.2.

Transforming to Discrete Time How the resulting linear differential equation models can be transformed into equivalent models in discrete time is the focus of Sections 2.3 and 2.4.

[3]The italicized terms in this sentence will be defined more precisely later.

Observability from Outputs Whether or not the state of a dynamic system model can be determined from its outputs is the focus of Section 2.5.

2.2 DYNAMIC SYSTEM MODELS

2.2.1 Dynamic Systems Modeled by Differential Equations

A *system* is an assemblage of interrelated entities that can be considered as a whole. If the attributes of interest of a system are changing with time, then it is called a *dynamic system*. A *process* is the evolution over time of a dynamic system.

Our *solar system*, consisting of the sun and its planets, is a physical example of a dynamic system. The motions of these bodies are governed by laws of motion that depend only upon their current relative positions and velocities. Sir Isaac Newton (1642–1727) discovered these laws and expressed them as a system of differential equations—another of his discoveries. From the time of Newton, engineers and scientists have learned to define dynamic systems in terms of the differential equations that govern their behavior. They have also learned how to solve many of these differential equations to obtain formulas for predicting the future behavior of dynamic systems.

Example 2.1 (Newton's Model for a Dynamic System of *n* Massive Bodies) For a planetary system with n bodies, as illustrated in Figure 2.2(a) for $n = 4$, the acceleration of the ith body in any inertial (i.e., nonrotating and nonaccelerating) Cartesian coordinate system is given by Newton's third law as the second-order differential equation

$$\frac{d^2 r_i}{dt^2} = C_g \sum_{\substack{j=1 \\ j \neq i}}^{n} \frac{m_j [r_j - r_i]}{|r_j - r_i|^3}, 1 \leq i \leq n,$$

where r_j is the position coordinate vector of the jth body, m_j is the mass of the jth body, and C_g is the gravitational constant. This set of n differential equations, plus

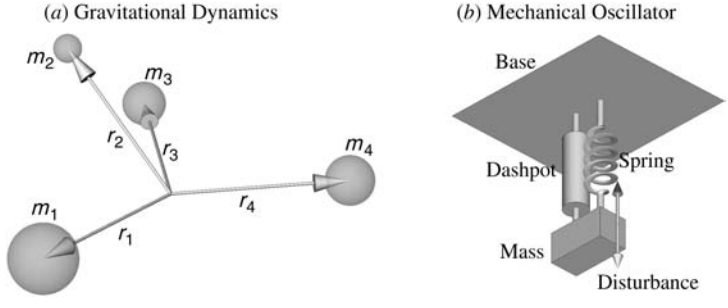


Figure 2.2 Dynamic system examples.

the associated initial conditions of the bodies (i.e., their initial positions and velocities), theoretically determines the future history of the planetary system.

This differential equation is homogeneous in the sense that it contains no terms that do not depend on the dependent variables r_j.

Example 2.2 (Harmonic Resonator with Linear Damping) Consider the idealized apparatus in Figure 2.2(*b*) with a mass m attached through a spring to an immovable base and its frictional contact to its support base represented by a dashpot. Let δ be the displacement of the mass from its position at rest, $d\delta/dt$ be the velocity of the mass, and $a(t) = d^2\delta/dt^2$ its acceleration. The force F acting on the mass can be represented by Newton's second law as

$$\begin{aligned} F(t) &= ma(t) \\ &= m\left[\frac{d^2\delta}{dt^2}(t)\right] \\ &= -k_s\delta(t) - k_d\frac{d\delta}{dt}(t), \end{aligned}$$

where k_s is the spring constant and k_d is the drag coefficient of the dashpot. This relationship can be written as a differential equation

$$m\frac{d^2\delta}{dt^2} = -k_s\delta - k_d\frac{d\delta}{dt}$$

in which time (t) is the differential variable and displacement (δ) is the dependent variable. This equation constrains the dynamic behavior of the damped harmonic resonator. The order of a differential equation is the order of the highest derivative, which is 2 in this example. This one is called a *linear differential equation*, because both sides of the equation are linear combinations of δ and its derivatives. (That of Example 2.1 is a *nonlinear differential equation*.)

Not all Dynamic Systems Can Be Modeled by Differential Equations There are other types of dynamic system models, such as Petri nets, inference nets, or tables of experimental data. However, the only types of dynamic systems considered in this book will be modeled by differential equations or by discrete-time linear state dynamic equations derived from linear differential or difference equations.

2.2.2 State Variables and State Equations

The second-order differential equation of the previous example can be transformed into a system of two first-order differential equations in the two dependent variables $x_1 = \delta$ and $x_2 = d\delta/dt$. In this way, one can reduce the form of any system of higher

order differential equations to an equivalent system of first-order differential equations. These systems are generally classified into the types shown in Table 2.1, with the most general type being a *time-varying* differential equation for representing a dynamic system with time-varying dynamic characteristics. This is represented in vector form as

$$\dot{x}(t) = f(t, x(t), u(t)), \tag{2.1}$$

where Newton's "dot" notation is used as a shorthand for the derivative with respect to time and a vector-valued function f to represent a system of n equations

$$\begin{aligned}
\dot{x}_1 &= f_1(t, x_1, x_2, x_3, \ldots, x_n, u_1, u_2, u_3, \ldots, u_r, t),\\
\dot{x}_2 &= f_2(t, x_1, x_2, x_3, \ldots, x_n, u_1, u_2, u_3, \ldots, u_r, t),\\
\dot{x}_3 &= f_3(t, x_1, x_2, x_3, \ldots, x_n, u_1, u_2, u_3, \ldots, u_r, t),\\
&\vdots\\
\dot{x}_n &= f_n(t, x_1, x_2, x_3, \ldots, x_n, u_1, u_2, u_3, \ldots, u_r, t)
\end{aligned} \tag{2.2}$$

in the independent variable t (time), n dependent variables $\{x_i | 1 \leq i \leq n\}$, and r known inputs $\{u_i | 1 \leq i \leq r\}$. These are called the *state equations* of the dynamic system.

2.2.2.1 Homogeneous and Nonhomogeneous Differential Equations The variables u_j in Equations (2.2) can be independent of the state variables x_j, in which case the system of equations is considered *nonhomogeneous.* This distinction can become a bit muddled in the context of control theory, however, where the u_j may become control inputs that are functions of the x_j (or of the dynamic system outputs, which are functions of the x_j).

In general, differential equations containing terms that do not depend on the dependent variables and their derivatives are called *nonhomogeneous*, and those without such nonhomogeneous terms (i.e., the u_j) are called *homogeneous.*

2.2.2.2 State Variables Represent the Degrees of Freedom of Dynamic Systems The variables $x_1, \ldots, x_n$ are called the *state variables* of the dynamic system defined by Equation 2.2. They are collected into a single n-vector

$$x(t) = [x_1(t) \;\; x_2(t) \;\; x_3(t) \;\; \cdots \;\; x_n(t)]^{\mathrm{T}} \tag{2.3}$$

called the *state vector* of the dynamic system. The n-dimensional domain of the state vector is called the *state space* of the dynamic system. Subject to certain continuity conditions on the functions f_i and u_i, the values $x_i(t_0)$ at some initial time t_0 will uniquely determine the values of the solutions $x_i(t)$ on some closed time interval $t \in [t_0, t_f]$ with initial time t_0 and final time t_f. In that sense, the initial value of each state variable represents an independent degree of freedom of the dynamic system.

The n values $x_1(t_0)$, $x_2(t_0)$, $x_3(t_0)$, ..., $x_n(t_0)$ can be varied independently, and they uniquely determine the state of the dynamic system over the time interval $t_0 \leq t \leq t_f$.

Example 2.3 (State Space Model of the Harmonic Resonator) For the second-order differential equation introduced in Example 2.2, let the state variables $x_1 = \delta$ and $x_2 = \dot{\delta}$. The first state variable represents the displacement of the mass from static equilibrium, and the second state variable represents the instantaneous velocity of the mass. The system of first-order differential equations for this dynamic system can be expressed in matrix form as

$$\frac{d}{dt}\begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = F_c \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix},$$

$$F_c = \begin{bmatrix} 0 & 1 \\ -\dfrac{k_s}{m} & -\dfrac{k_d}{m} \end{bmatrix},$$

where F_c is called the *coefficient matrix* of the system of first-order linear differential equations. This is an example of what is called the *companion form* for higher order linear differential equations expressed as a system of first-order differential equations.

2.2.3 Continuous Time and Discrete Time

The dynamic system defined by Equation (2.2) is an example of a *continuous* system, so called because it is defined with respect to an independent variable t that varies continuously over some real interval $t \in [t_0, t_f]$. For many practical problems, however, one is only interested in knowing the state of a system at a discrete set of times $t \in \{t_1, t_2, t_3, \ldots\}$. These discrete times may, for example, correspond to the times at which the outputs of a system are sampled (such as the times at which Piazzi recorded the direction to Ceres). For problems of this type, it is convenient to order the times t_k according to their integer subscripts:

$$t_0 < t_1 < t_2 < \cdots t_{k-1} < t_k < t_{k+1} < \cdots.$$

That is, the time sequence is ordered according to the subscripts, and the subscripts take on all successive values in some range of integers. For problems of this type, it suffices to define the state of the dynamic system as a recursive relation,

$$x(t_{k+1}) = f\,(x(t_k), t_k, t_{k+1}), \tag{2.4}$$

by means of which the state is represented as a function of its previous state. This is a definition of a *discrete dynamic system*. For systems with *uniform time intervals* Δt

$$t_k = k\Delta t.$$

*2.2.3.1 **Shorthand Notation for Discrete-Time Systems*** It uses up a lot of ink if one writes $x(t_k)$ when all one cares about is the sequence of values of the state variable x. It is more efficient to shorten this to x_k, so long as it is understood that it stands for $x(t_k)$, and not the kth component of x. If one must talk about a particular component at a particular time, one can always resort to writing $x_i(t_k)$ to remove any ambiguity. Otherwise, let us drop t as a symbol whenever it is clear from the context that we are talking about discrete-time systems.

2.2.4 Time-Varying Systems and Time-Invariant Systems

The term *physical plant* or *plant* is sometimes used in place of *dynamic system*, especially for applications in manufacturing. In many such applications, the dynamic system under consideration is literally a physical plant—a fixed facility used in the manufacture of materials. Although the input $u(t)$ may be a function of time, the functional dependence of the state dynamics on u and x does not depend upon time. Such systems are called *time-invariant* or *autonomous*. Their solutions are generally easier to obtain than those of time-varying systems.

2.3 CONTINUOUS LINEAR SYSTEMS AND THEIR SOLUTIONS

2.3.1 Input/Output Models of Linear Dynamic Systems

The blocks labeled "STATE DYNAMICS" and "SENSOR MODEL" in Figure 2.1 model a dynamic system with the equation models listed in Table 2.1 and Figure 2.3. These models represent dynamic systems using five types of variables:

- *Control inputs* u_j, which can be under our control and therefore known to us.
- *Random dynamic disturbances* w_j, which are known only statistically (i.e., with known statistical properties).
- Internal *state variables* x_j, which were described in the previous section. In most applications, these are "hidden" variables in the sense that they

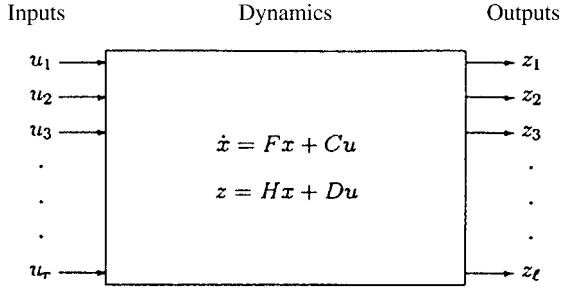


Figure 2.3 Block diagram of a linear dynamic system.

cannot generally be measured directly but must be inferred from what can be measured.

- *Outputs* z_j, also called *observations* or *measurements*, which are those things that can be known from sensors used for measuring some of the internal state variables x_j.
- *Sensor noise* v_j, also known only statistically (i.e., with known statistical properties).

These concepts are discussed in greater detail in the following subsections.

2.3.2 Dynamic Coefficient Matrices and Input Coupling Matrices

The dynamics of linear systems are represented by a set of n first-order linear differential equations expressible in vector form as

$$\begin{aligned}\dot{x}(t) &= \frac{d}{dt}x(t) \\ &= F(t)x(t) + C(t)u(t),\end{aligned} \tag{2.5}$$

where the elements and components of the matrices and vectors can be functions of time:

$$F(t) = \begin{bmatrix} f_{11}(t) & f_{12}(t) & f_{13}(t) & \cdots & f_{1n}(t) \\ f_{21}(t) & f_{22}(t) & f_{23}(t) & \cdots & f_{2n}(t) \\ f_{31}(t) & f_{32}(t) & f_{33}(t) & \cdots & f_{3n}(t) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ f_{n1}(t) & f_{n2}(t) & f_{n3}(t) & \cdots & f_{nn}(t) \end{bmatrix},$$

$$C(t) = \begin{bmatrix} c_{11}(t) & c_{12}(t) & c_{13}(t) & \cdots & c_{1r}(t) \\ c_{21}(t) & c_{22}(t) & c_{23}(t) & \cdots & c_{2r}(t) \\ c_{31}(t) & c_{32}(t) & c_{33}(t) & \cdots & c_{3r}(t) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_{n1}(t) & c_{n2}(t) & c_{n3}(t) & \cdots & c_{nr}(t) \end{bmatrix},$$

$$u(t) = \begin{bmatrix} u_1(t) & u_2(t) & u_3(t) & \cdots & u_r(t) \end{bmatrix}^{\mathrm{T}}.$$

The matrix $F(t)$ is called the *dynamic coefficient matrix* or simply the *dynamic matrix*. Its elements are called the *dynamic coefficients*. The matrix $C(t)$ is called the *input coupling matrix*, and its elements are called *input coupling coefficients*. The r-vector u is called the *input vector*.

Nonhomogeneous Part The term $C(t)u(t)$ in Equation 2.5 is called the *nonhomogeneous part* of the differential equation, because it does not depend upon the dependent variable x.

Example 2.4 (Dynamic Equation for a Heating/Cooling System) Consider the temperature T in a heated, enclosed room or building as the state variable of a dynamic system. A simplified plant model for this dynamic system is the linear equation

$$\dot{T}(t) = -k_c[T(t) - T_o(t)] + k_h u(t),$$

where the constant "cooling coefficient" k_c depends on the quality of thermal insulation from the outside, T_o is the temperature outside, k_h is the heating/cooling rate coefficient of the heater or cooler, and u is an input function that is either $u = 0$ (off) or $u = 1$ (on) and can be defined as a function of any measurable quantities. The outside temperature T_o, on the other hand, is an example of an input function which may be directly measurable at any time but is not predictable in the future. It is effectively a random process.

2.3.3 Companion Form for Higher Order Derivatives

In general, the nth-order linear differential equation

$$\frac{d^n y(t)}{dt^n} + f_1(t)\frac{d^{n-1}y(t)}{dt^{n-1}} + \cdots + f_{n-1}(t)\frac{dy(t)}{dt} + f_n(t) = u(t) \tag{2.6}$$

can be rewritten as a system of n first-order differential equations. Although the state variable representation as a first-order system is not unique [47], there is a unique way of representing it called the *companion form*.

2.3.3.1 Companion Form of the State Vector For the nth-order linear dynamic system shown above, the companion form of the state vector is

$$x(t) = \left[y(t), \quad \frac{d}{dt}y(t), \quad \frac{d^2}{dt^2}y(t), \ldots, \frac{d^{n-1}}{dt^{n-1}}y(t) \right]^{\mathrm{T}}. \tag{2.7}$$

2.3.3.2 Companion Form of the Differential Equation The nth-order linear differential equation can be rewritten in terms of the above state vector $x(t)$ as the

vector differential equation

$$\frac{d}{dt}\begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_{n-1}(t) \\ x_n(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ -f_n(t) & -f_{n-1}(t) & -f_{n-2}(t) & \cdots & -f_1(t) \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ \vdots \\ x_n(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} u(t). \tag{2.8}$$

When Equation 2.8 is compared with Equation 2.5, the matrices $F(t)$ and $C(t)$ are easily identified.

Displacement Matrices The upper-left $(n-1) \times (n-1)$ submatrix of the companion matrix F of Equation 2.8 is the transpose of what is called a *displacement matrix*,

$$D = \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix}, \tag{2.9}$$

which has come to play an important role in the development of parallel processing methods for many matrix operations, including some in Kalman filtering [138].

2.3.3.3 The Companion Form Is Ill-Conditioned Although it simplifies the relationship between higher order linear differential equations and first-order systems of differential equations, the companion matrix is not recommended for implementation. Studies by Kenney and Liepnik [152] have shown that it is poorly conditioned for solving differential equations.

2.3.4 Outputs and Measurement Sensitivity Matrices

2.3.4.1 Measurable Outputs and Measurement Sensitivities Only the inputs and outputs of the system can be measured, and it is usual practice to consider the variables z_i as the measured values. For linear problems, they are related to the

state variables and the inputs by a system of linear equations that can be represented in vector form as

$$z(t) = H(t)x(t) + D(t)u(t), \tag{2.10}$$

where

$$z(t) = \begin{bmatrix} z_1(t) & z_2(t) & z_3(t) & \cdots & z_\ell(t) \end{bmatrix}^{\mathrm{T}},$$

$$H(t) = \begin{bmatrix} h_{11}(t) & h_{12}(t) & h_{13}(t) & \cdots & h_{1n}(t) \\ h_{21}(t) & h_{22}(t) & h_{23}(t) & \cdots & h_{2n}(t) \\ h_{31}(t) & h_{32}(t) & h_{33}(t) & \cdots & h_{3n}(t) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ h_{\ell 1}(t) & h_{\ell 2}(t) & h_{\ell 3}(t) & \cdots & h_{\ell n}(t) \end{bmatrix},$$

$$D(t) = \begin{bmatrix} d_{11}(t) & d_{12}(t) & d_{13}(t) & \cdots & d_{1r}(t) \\ d_{21}(t) & d_{22}(t) & d_{23}(t) & \cdots & d_{2r}(t) \\ d_{31}(t) & d_{32}(t) & d_{33}(t) & \cdots & d_{3r}(t) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ d_{\ell 1}(t) & d_{\ell 2}(t) & d_{\ell 3}(t) & \cdots & d_{\ell r}(t) \end{bmatrix}.$$

The ℓ-vector $z(t)$ is called the *measurement vector* or the *output vector* of the system. The coefficient $h_{ij}(t)$ represents the *sensitivity* (measurement sensor scale factor) of the ith measured output to the jth internal state. The matrix $H(t)$ of these values is called the *measurement sensitivity matrix*, and $D(t)$ is called the *input/output coupling matrix*. The *measurement sensitivities* $h_{ij}(t)$ and input/output coupling coefficients $d_{ij}(t)$, $1 \le i \le \ell$, $1 \le j \le r$ are known functions of time. The state equation, 2.5, and the output equation, 2.10, together form the dynamic equations of the system shown in Table 2.1.

2.3.5 Difference Equations and State Transition Matrices (STM)

Difference equations are the discrete-time versions of differential equations. They are usually written in terms of *forward differences* $x(t_{k+1}) - x(t_k)$ of the state variable (the dependent variable), expressed as a function ψ of all independent variables or of the forward value $x(t_{k+1})$ as a function ϕ of all independent variables (including the previous value as an independent variable):

$$x(t_{k+1}) - x(t_k) = \psi(t_k, x(t_k), u(t_k)),$$

or

$$x(t_{k+1}) = \phi(t_k, x(t_k), u(t_k)), \tag{2.11}$$

$$\phi(t_k, x(t_k), u(t_k)) = x(t_k) + \psi(t_k, x(t_k), u(t_k)).$$

The second of these (Equation 2.11) has the same general form of the recursive relation shown in Equation 2.4, which is the one that is usually implemented for discrete-time systems.

For linear dynamic systems, the functional dependence of $x(t_{k+1})$ on $x(t_k)$ and $u(t_k)$ can be represented by matrices:

$$\begin{aligned} x(t_{k+1}) - x(t_k) &= \Psi(t_k)x(t_k) + C(t_k)u(t_k), \\ x_{k+1} &= \Phi_k x_k + C_k u_k, \\ \Phi_k &= I + \Psi(t_k), \end{aligned} \tag{2.12}$$

where the matrices Ψ and Φ replace the functions ψ and ϕ, respectively. The matrix Φ is called the *state transition matrix*. The matrix C is called the *discrete-time input coupling matrix* or simply the *input coupling matrix*—if the discrete-time context is already established.

2.3.6 Solving Differential Equations for STMs

A state transition matrix is a solution of what is called the *homogeneous*[4] matrix equation associated with a given linear dynamic system. Let us define homogeneous equations and then show how their solutions are related to the solutions of a given linear dynamic system.

2.3.6.1 Homogeneous Systems The equation $\dot{x}(t) = F(t)x(t)$ is called the *homogeneous part* of the linear differential equation $\dot{x}(t) = F(t)x(t) + C(t)u(t)$. The solution of the homogeneous part can be obtained more easily than that of the full equation, and its solution is used to define the solution to the general (nonhomogeneous) linear equation.

2.3.6.2 Fundamental Solutions of Homogeneous Equations An $n \times n$ matrix-valued function $\Phi(t)$ is called a *fundamental solution* of the homogeneous equation $\dot{x}(t) = F(t)x(t)$ on the interval $t \in [0, T]$ if $\dot{\Phi}(t) = F(t)\Phi(t)$ and $\Phi(0) = I_n$, the $n \times n$ identity matrix. Note that, for any possible initial vector $x(0)$, the vector $x(t) = \Phi(t)x(0)$ satisfies the equation

$$\dot{x}(t) = \frac{d}{dt}[\Phi(t)x(0)] \tag{2.13}$$

$$= \left[\frac{d}{dt}\Phi(t)\right]x(0) \tag{2.14}$$

$$= [F(t)\Phi(t)]x(0) \tag{2.15}$$

$$= F(t)[\Phi(t)x(0)] \tag{2.16}$$

$$= F(t)x(t). \tag{2.17}$$

[4]This terminology comes from the notion that every term in the expression so labeled contains the dependent variable. That is, the expression is *homogeneous* with respect to the dependent variable.

That is, $x(t) = \Phi(t)x(0)$ is the solution of the homogeneous equation $\dot{x} = Fx$ with initial value $x(0)$.

Example 2.5 (STM for Homogeneous System) The unit upper triangular Toeplitz matrix

$$\Phi(t) = \begin{bmatrix} 1 & t & \frac{1}{2}t^2 & \frac{1}{1\cdot 2\cdot 3}t^3 & \cdots & \frac{1}{(n-1)!}t^{n-1} \\ 0 & 1 & t & \frac{1}{2}t^2 & \cdots & \frac{1}{(n-2)!}t^{n-2} \\ 0 & 0 & 1 & t & \cdots & \frac{1}{(n-3)!}t^{n-3} \\ 0 & 0 & 0 & 1 & \cdots & \frac{1}{(n-4)!}t^{n-4} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 \end{bmatrix}$$

is the fundamental solution of $\dot{x} = Fx$ for the strictly upper triangular Toeplitz dynamic coefficient matrix

$$F = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix},$$

which can be verified by showing that $\Phi(0) = I$ and $\dot{\Phi} = F\Phi$. This dynamic coefficient matrix, in turn, is the companion matrix for the nth-order linear homogeneous differential equation $(d/dt)^n y(t) = 0$.

2.3.6.3 Existence and Nonsingularity of Fundamental Solutions If the elements of the matrix $F(t)$ are continuous functions on some interval $0 \le t \le T$, then the fundamental solution matrix $\Phi(t)$ is guaranteed to exist and to be nonsingular on an interval $0 \le t \le \tau$ for some $\tau > 0$. These conditions also guarantee that $\Phi(t)$ will be nonsingular on some interval of nonzero length as a consequence of the continuous dependence of the solution $\Phi(t)$ of the matrix equation on its (nonsingular) initial conditions [$\Phi(0) = I$] [66].

2.3.6.4 STMs Note that the fundamental solution matrix $\Phi(t)$ transforms any initial state $x(0)$ of the dynamic system into the corresponding state $x(t)$ at time t. If $\Phi(t)$ is nonsingular, then the products $\Phi^{-1}(t)x(t) = x(0)$ and $\Phi(\tau)\Phi^{-1}(t)x(t) = x(\tau)$. That is, the matrix product

$$\Phi(\tau, t) = \Phi(\tau)\Phi^{-1}(t) \tag{2.18}$$

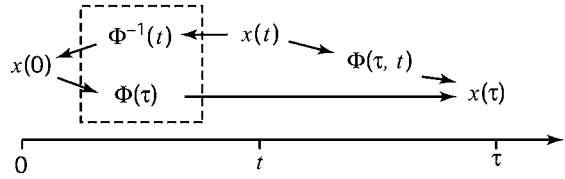


Figure 2.4 The STM as a composition of fundamental solution matrices.

transforms a solution from time t into the corresponding solution at time τ, as diagrammed in Figure 2.4. Such a matrix is called the evolution operator[5] for the associated linear homogeneous differential equation. The state transition matrix $\Phi(\tau, t)$ represents the transition to the state at time τ from the state at time t.

2.3.6.5 Properties of STMs and Fundamental Solution Matrices The same symbol (Φ) has been used for *fundamental solution matrices* and for STMs, the distinction being made by the number of arguments. By convention, then,

$$\Phi(\tau, 0) = \Phi(\tau).$$

Other useful properties of Φ include the following:

1. $\Phi(\tau, \tau) = \Phi(0) = I$.
2. $\Phi^{-1}(\tau, t) = \Phi(t, \tau)$.
3. $\Phi(\tau, \sigma)\Phi(\sigma, t) = \Phi(\tau, t)$.
4. $(\partial/\partial\tau)\Phi(\tau, t) = F(\tau)\Phi(\tau, t)$.
5. $(\partial/\partial t)\Phi(\tau, t) = -\Phi(\tau, t)F(t)$.

Example 2.6 (STM for the Underdamped Harmonic Resonator Model) The general solution of the differential equation. In Examples 2.2 and 2.3, the displacement δ of the damped harmonic resonator was modeled by the state equation

$$x = \begin{bmatrix} \delta \\ \dot{\delta} \end{bmatrix},$$

$$\dot{x} = Fx,$$

$$F = \begin{bmatrix} 0 & 1 \\ -\dfrac{k_s}{m} & -\dfrac{k_d}{m} \end{bmatrix}.$$

[5]Formally, an operator $\Phi(t, t_0, x(t_0))$ such that $x(t) = \Phi(t, t_0, x(t_0))$ is called an *evolution operator* for a dynamic system with state x. A STM is a linear evolution operator.

The characteristic values of the dynamic coefficient matrix F are the roots of its characteristic polynomial

$$det(\lambda I - F) = \lambda^2 + \frac{k_d}{m}\lambda + \frac{k_s}{m},$$

which is a quadratic polynomial with roots

$$\lambda_1 = \frac{1}{2}\left(-\frac{k_d}{m} + \sqrt{\frac{k_d^2}{m^2} - \frac{4k_s}{m}}\right),$$

$$\lambda_2 = \frac{1}{2}\left(-\frac{k_d}{m} - \sqrt{\frac{k_d^2}{m^2} - \frac{4k_s}{m}}\right).$$

The general solution for the displacement δ can then be written in the form

$$\delta(t) = \alpha e^{\lambda_1 t} + \beta e^{\lambda_2 t},$$

where α and β are (possibly complex) free variables.

The underdamped solution. The resonator is considered underdamped if the discriminant

$$\frac{k_d^2}{m^2} - \frac{4k_s}{m} < 0,$$

in which case the roots are a conjugate pair of nonreal complex numbers and the general solution can be rewritten in "real form" as

$$\delta(t) = ae^{-t/\tau}\cos(\omega t) + be^{-t/\tau}\sin(\omega t),$$

$$\tau = \frac{2m}{k_d},$$

$$\omega = \sqrt{\frac{k_s}{m} - \frac{k_d^2}{4m^2}},$$

where a and b are now real variables, τ is the decay time constant, and ω is the resonator resonant frequency. This solution can be expressed in state-space form in terms of the real variables a and b:

$$\begin{bmatrix} \delta(t) \\ \dot{\delta}(t) \end{bmatrix} = e^{-t/\tau}\begin{bmatrix} \cos(\omega t) & \sin(\omega t) \\ -\dfrac{\cos(\omega t)}{\tau} - \omega\sin(\omega t) & \omega\cos(\omega t) - \dfrac{\sin(\omega t)}{\tau} \end{bmatrix}\begin{bmatrix} a \\ b \end{bmatrix}.$$

Initial value constraints. The initial values

$$\delta(0) = a, \quad \dot{\delta}(0) = -\frac{a}{\tau} + \omega b$$

can be solved for a and b as

$$\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \frac{1}{\omega\tau} & \frac{1}{\omega} \end{bmatrix} \begin{bmatrix} \delta(0) \\ \dot{\delta}(0) \end{bmatrix}.$$

This can then be combined with the solution for $x(t)$ in terms of a and b to yield the fundamental solution

$$x(t) = \Phi(t)x(0),$$

$$\Phi(t) = \frac{e^{-t/\tau}}{\omega\tau^2} \begin{bmatrix} \tau[\omega\tau\cos(\omega t) + \sin(\omega t)] & \tau^2 \sin(\omega t) \\ -(1+\omega^2\tau^2)\sin(\omega\tau) & [\omega\tau^2\cos(\omega t) - \tau\sin(\omega t)] \end{bmatrix}$$

in terms of the damping time constant and the resonant frequency.

2.3.7 Solution of Nonhomogeneous Equations

The solution of the nonhomogeneous state Equation 2.5 is given by

$$x(t) = \Phi(t, t_0)x(t_0) + \int_{t_0}^{t} \Phi(t,\tau)C(\tau)u(\tau)\,d\tau \tag{2.19}$$

$$= \Phi(t)\Phi^{-1}(t_0)x(t_0) + \Phi(t)\int_{t_0}^{t} \Phi^{-1}(\tau)C(\tau)u(\tau)\,d\tau, \tag{2.20}$$

where $x(t_0)$ is the initial value and $\Phi(t, t_0)$ is the STM of the dynamic system defined by $F(t)$. (This can be verified by taking derivatives and using the properties of STM given above.)

2.3.8 Closed-Form Solutions of Time-Invariant Systems

2.3.8.1 Using Matrix Exponentials In this case, the coefficient matrix F is a constant function of time. The solution will still be a function of time, but the associated state transition matrices $\Phi(t, \tau)$ will depend only on the differences $t-\tau$. In fact, one can show that

$$\Phi(t, \tau) = e^{F(t-\tau)} \tag{2.21}$$

$$= \sum_{i=0}^{\infty} \frac{(t-\tau)^i}{i!} F^i, \tag{2.22}$$

where $F^0 = I$ by definition. Equation 2.21 uses the matrix exponential function defined by Equation 2.22.

The solution of the nonhomogeneous equation in this case will be

$$x(t) = e^{F(t-\tau)}x(\tau) + \int_{\tau}^{t} e^{F(t-\sigma)}Cu(\sigma)\,d\sigma \tag{2.23}$$

$$= e^{F(t-\tau)}x(\tau) + e^{Ft}\int_{\tau}^{t} e^{-F\sigma}Cu(\sigma)\,d\sigma. \tag{2.24}$$

2.3.8.2 Using Laplace Transforms Equation 2.22 can be used to derive the state transition matrix $\Phi(t)$ using inverse Laplace transforms. The Laplace transform of $\Phi(t) = \exp(Ft)$ will be

$$\begin{aligned}
\mathcal{L}\Phi(t) &= \mathcal{L}[\exp(Ft)] \\
&= \mathcal{L}\left[\sum_{k=0}^{\infty}\frac{1}{k!}F^k t^k\right] \\
&= \sum_{k=0}^{\infty}\frac{1}{k!}F^k\left[\mathcal{L}t^k\right] \\
&= \sum_{k=0}^{\infty}\frac{1}{k!}F^k\left[\frac{k!}{s^{k+1}}\right] \\
&= \frac{1}{s}\sum_{k=0}^{\infty}\left(s^{-1}F\right)^k \\
&= \frac{1}{s}\left[I - s^{-1}F\right]^{-1} \\
&= [sI - F]^{-1},
\end{aligned} \tag{2.25}$$

and consequently, $\Phi(t)$ can be derived analytically in terms of the inverse Laplace transform $\mathcal{L}^{-1}$ as

$$\Phi(t) = \mathcal{L}^{-1}\left\{[sI - F]^{-1}\right\}. \tag{2.26}$$

Equations 2.22 (matrix exponential) and 2.26 (inverse Laplace transform) can be used to derive the STM in "closed form" (i.e., as a formula) for dynamic systems modeled by linear time-invariant differential equations.

2.3.8.3 Computing Matrix Exponentials The following methods have been used for computing matrix exponentials numerically:

1. A *scaling and squaring* method combined with a Padé approximation is the recommended general-purpose method. This method is discussed in greater

detail in Section B.4.4.2. The MATLAB function `expm` uses this method to compute the matrix exponential numerically.

2. Numerical integration of the homogeneous part of the differential equation,

$$\frac{d}{dt}\Phi(t) = F\Phi(t), \tag{2.27}$$

with initial value $\Phi(0) = I$. This method also works for time-varying systems.

3. The approximation of e^{Ft} by a truncated power series expansion is *not* a recommended general-purpose method. Its convergence is poor unless the characteristic values of Ft are well inside the unit circle in the complex plane.

There are many other methods for computing matrix exponentials,[6] but these are the most important.

Example 2.7 (Laplace Transform Method) We will now demonstrate a Laplace transform method for deriving the state transition matrix for nonhomogeneus differential equations, using the damped resonator model of Examples 2.2, 2.3, and 2.6 with a specified set of model parameter values. The result is a formula for the fundamental solution for a damped harmonic oscillator with this particular set of model parameters.

Differential Equation Model This model can be written in the form of a second-order differential equation

$$\ddot{\delta}(t) + 2\zeta\omega_n\dot{\delta}(t) + \omega_n^2\delta(t) = u(t),$$

where

$$\dot{\delta}(t) = \frac{d\delta}{dt}, \quad \ddot{\delta}(t) = \frac{d^2\delta}{dt^2}, \quad \zeta = \frac{k_d}{2\sqrt{mk_s}}, \quad \omega_n = \sqrt{\frac{k_s}{m}}.$$

The parameter ζ is a unitless damping coefficient, and ω_n is the "natural" (i.e., undamped) frequency of the resonator.

State-Space Form This second-order linear differential equation can be rewritten in a state-space form, with states $x_1 = \delta$ and $x_2 = \dot{\delta} = \dot{x}_1$ and parameters ζ and ω_n, as

$$\frac{d}{dt}\begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 \\ -\omega_n^2 & -2\zeta\omega_n \end{bmatrix}}_{F}\begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ 1 \end{bmatrix}}_{C} u(t)$$

[6] See, for example, Brockett [47], DeRusso et al. [72], or Kreindler and Sarachik [156].

with initial conditions

$$x(t_0) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

and

$$u(t) \equiv 1, \quad \omega_n = 1, \quad \zeta = 0.5,$$

so that the matrices

$$F = \begin{bmatrix} 0 & 1 \\ -1 & -1 \end{bmatrix} \qquad Cu(t) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

STM As an alternative to using the matrix exponential, one can obtain the state transition matrix by applying the inverse Laplace transform $\mathcal{L}^{-1}$ to the Laplace transformed matrix $sI - F$:

$$(sI - F) = \begin{bmatrix} s & -1 \\ 1 & s+1 \end{bmatrix},$$

$$(sI - F)^{-1} = \frac{1}{s^2 + s + 1} \begin{bmatrix} s+1 & 1 \\ -1 & s \end{bmatrix}$$

$$\Phi(t) = \mathcal{L}^{-1}(sI - F)^{-1}$$

$$= \mathcal{L}^{-1} \begin{bmatrix} \dfrac{s+1}{s^2+s+1} & \dfrac{1}{s^2+s+1} \\ \dfrac{-1}{s^2+s+1} & \dfrac{s}{s^2+s+1} \end{bmatrix}$$

$$= \frac{2e^{-t/2}}{\sqrt{3}} \begin{bmatrix} \frac{\sqrt{3}}{2}\cos(\omega_d) + \frac{1}{2}\sin(\omega_d) & \sin(\omega_d) \\ -\sin(\omega_d) & \frac{\sqrt{3}}{2}\cos(\omega_d) - \frac{1}{2}\sin(\omega_d) \end{bmatrix}$$

$$\omega_d = \frac{\sqrt{3}}{2} \quad \omega_n = \frac{\sqrt{3}}{2}$$

General Solution Equation 2.24 with $\tau = 0$ becomes

$$x(t) = \Phi(t)x(0) + \Phi(t) \int_0^t \Phi^{-1}(\tau) Cu(\tau)\, d\tau,$$

where, in this example,

$$\Phi^{-1}(\tau) = \frac{e^{\tau/2}}{3} \begin{bmatrix} 3\cos(\omega_d) - \sqrt{3}\sin(\omega_d) & -2\sqrt{3}\sin(\omega_d) \\ 2\sqrt{3}\sin(\omega_d) & 3\cos(\omega_d) + \sqrt{3}\sin(\omega_d) \end{bmatrix}$$

$$\Phi^{-1}(\tau) Cu(\tau) = \frac{e^{\tau/2}}{3} \begin{bmatrix} -2\sqrt{3}\sin(\omega_d) \\ 3\cos(\omega_d) + \sqrt{3}\sin(\omega_d) \end{bmatrix}$$

$$\int_0^t \Phi^{-1}(\tau)Cu(\tau) = \begin{bmatrix} e^{t/2}\cos(\omega_d) - \frac{1}{3}\sqrt{3}e^{t/2}\sin(\omega_d) - 1 \\ \frac{2}{3}\sqrt{3}e^{t/2}\sin(\omega_d) \end{bmatrix}$$

$$\Phi(t)\int_0^t \Phi^{-1}(\tau)Cu(\tau) = \begin{bmatrix} -e^{-t/2}\cos(\omega_d) - \frac{1}{3}\sqrt{3}e^{-t/2}\sin(\omega_d) + 1 \\ \frac{2}{3}\sqrt{3}e^{-t/2}\sin(\omega_d) \end{bmatrix}$$

$$\begin{aligned} x(t) &= \Phi(t(0) + \Phi(t)\int_0^t \Phi^{-1}(\tau)Cu(\tau)\,d\tau \\ &= \Phi(t)\begin{bmatrix} 0 \\ 1 \end{bmatrix} + \begin{bmatrix} -e^{-t/2}\cos(\omega_d) - \frac{1}{3}\sqrt{3}e^{-t/2}\sin(\omega_d) + 1 \\ \frac{2}{3}\sqrt{3}e^{-t/2}\sin(\omega_d) \end{bmatrix} \\ &= \frac{1}{3}e^{-t/2}\begin{bmatrix} 2\sqrt{3}\sin(\omega_d) \\ 3\cos(\omega_d) - \sqrt{3}\sin(\omega_d) \end{bmatrix} \\ &\quad + \begin{bmatrix} -e^{-t/2}\cos(\omega_d) - \frac{1}{3}\sqrt{3}e^{-t/2}\sin(\omega_d) + 1 \\ \frac{2}{3}\sqrt{3}e^{-t/2}\sin(\omega_d) \end{bmatrix} \end{aligned}$$

is the general solution. Note that all the sinusoidal terms die off exponentially, leaving the steady-state value

$$\lim_{t\to+\infty} x(t) = \begin{bmatrix} 1 \\ 0 \end{bmatrix},$$

which is due to the constant input $u(t) = 1$.

2.3.9 Time-Varying Systems

If $F(t)$ is not constant, the dynamic system is called *time-varying*. If $F(t)$ is a piecewise smooth function of t, the $n \times n$ homogeneous matrix differential equation 2.27 can be solved numerically by the fourth-order Runge–Kutta method.[7]

2.4 DISCRETE LINEAR SYSTEMS AND THEIR SOLUTIONS

2.4.1 Discretized Linear Systems

If one is only interested in the system state at discrete times, then one can use the formula

$$x(t_k) = \Phi(t_k, t_{k-1})x(t_{k-1}) + \int_{t_{k-1}}^{t_k} \Phi(t_k,\sigma)C(\sigma)u(\sigma)\,d\sigma \tag{2.28}$$

to propagate the state vector between the times of interest.

[7]Named after the German mathematicians Karl David Tolme Runge (1856–1927) and Wilhelm Martin Kutta (1867–1944).

2.4.1.1 Simplification for Constant u If u is constant over the interval $[t_{k-1}, t_k]$, then the above integral can be simplified to the form

$$x(t_k) = \Phi(t_k, t_{k-1})x(t_{k-1}) + \Gamma(t_{k-1})u(t_{k-1}) \tag{2.29}$$

$$\Gamma(t_{k-1}) = \int_{t_{k-1}}^{t_k} \Phi(t_k, \sigma)C(\sigma)\, d\sigma. \tag{2.30}$$

2.4.1.2 Shorthand Discrete-Time Notation For discrete-time systems, the indices k in the time sequence $\{t_k\}$ characterize the times of interest. One can save some ink by using the shorthand notation

$$x_k \stackrel{\text{def}}{=} x(t_k), \qquad z_k \stackrel{\text{def}}{=} z(t_k), \qquad u_k \stackrel{\text{def}}{=} u(t_k), \quad H_k \stackrel{\text{def}}{=} H(t_k),$$
$$D_k \stackrel{\text{def}}{=} D(t_k), \qquad \Phi_{k-1} \stackrel{\text{def}}{=} \Phi(t_k, t_{k-1}), \quad \Gamma_k \stackrel{\text{def}}{=} \Gamma(t_k)$$

for discrete-time systems, eliminating t entirely. Using this notation, one can represent the discrete-time state equations in the more compact form

$$x_k = \Phi_{k-1}x_{k-1} + \Gamma_{k-1}u_{k-1}, \tag{2.31}$$

$$z_k = H_k x_k + D_k u_k. \tag{2.32}$$

2.4.2 Discrete-Time Solution for Time-Invariant Systems

For continuous time-invariant systems that have been discretized using fixed time intervals, the matrices Φ, Γ, H, u, and D in Equations 2.31 and 2.32 are independent of the discrete-time index k as well. In that case, the solution can be written in closed form as

$$x_k = \Phi^k x_0 + \sum_{i=0}^{k-1} \Phi^{k-i-1}\Gamma u_i, \tag{2.33}$$

where Φ^k is the kth power of Φ.

Inverse z*-Transform Solution* The matrix powers Φ^k in Equation 2.33 can be computed using z-transforms as

$$\Phi^k = \left\{\mathcal{Z}^{-1}[(zI - \Phi)^{-1}z]\right\}_k, \tag{2.34}$$

where z is the z-transform variable and $\mathcal{Z}^{-1}$ is the inverse z-transform.

This is, in some ways, analogous to the inverse Laplace transform method used in Section 2.3.8.2 and Example 2.7 to compute the state transition matrix $\Phi(t)$ for linear time-invariant systems in continuous time. The analogous formulas for the inverse

transforms are

$$\begin{aligned}\Phi(t) &= \mathcal{L}^{-}1(sI - F)^{-1} \text{ (inverse Laplace transform)}\\ \Phi^k &= \left\{\mathcal{Z}^{-1}[(zI - F)^{-1}z]\right\}_k \text{ (inverse } z\text{-transform)}\\ &= \sum \text{residue of } [(zI - F)^{-1}z^k] \text{ (Cauchy residue theorem)}\end{aligned}$$

Example 2.8 (Solution Using z-Transforms) Consider Equation 2.33 with

$$x_0 = \begin{bmatrix}1\\0\end{bmatrix}, \qquad \Phi = \begin{bmatrix}0 & 1\\ -6 & -5\end{bmatrix}, \qquad \Gamma u = \begin{bmatrix}0\\1\end{bmatrix}.$$

The matrices are

$$\begin{aligned}[zI - \Phi]^{-1} &= \begin{bmatrix}z & -1\\ 6 & z+5\end{bmatrix}^{-1}\\ &= \begin{bmatrix}\dfrac{z+5}{(z+2)(z+3)} & \dfrac{1}{(z+2)(z+3)}\\ \dfrac{-6}{(z+2)(z+3)} & \dfrac{z}{(z+2)(z+3)}\end{bmatrix}\\ \sum \text{residue } [zI - \Phi]^{-1}z^k &= \begin{bmatrix}\dfrac{z^{k+1} + 5z^k}{(z+2)(z+3)} & \dfrac{z^k}{(z+2)(z+3)}\\ \dfrac{-6z^k}{(z+2)(z+3)} & \dfrac{z^{k+1}}{(z+2)(z+3)}\end{bmatrix}\\ \left\{\mathcal{Z}^{-1}[(zI - \Phi)^{-1}z]\right\}_k &= \begin{bmatrix}3\,(-2)^k - 2\,(-3)^k & (-2)^k - (-3)^k\\ -6\,[(-2)^k - (-3)^k] & -2\,(-2)^k + 3\,(-3)^k\end{bmatrix}\\ &= \Phi^k,\end{aligned}$$

so that the general solution of Equation 2.33 with the given values for x_0, Φ and Γu becomes

$$x_k = \begin{bmatrix}3\,(-2)^k - 2\,(-3)^k\\ -6\left[(-2)^k - (-3)^k\right]\end{bmatrix} + \sum_{i=0}^{k-1}\begin{bmatrix}(-2)^{k-i-1} - (-3)^{k-i-1}\\ -2\,(-2)^{k-i-1} + 3\,(-3)^{k-i-1}\end{bmatrix}.$$

2.5 OBSERVABILITY OF LINEAR DYNAMIC SYSTEM MODELS

Observability is the issue of whether the state of a dynamic system *with a known model* is uniquely determinable from its inputs and outputs. It is essentially a property of the given *system model*. A given linear dynamic system model with a given linear

input/output model is considered *observable* if and only if its state is *uniquely* determinable from the model's definition, its inputs, and its outputs. If the system state is *not* uniquely determinable from the system inputs and outputs, then the system model is considered *unobservable*.

How to Determine Whether a Given Dynamic System Model Is Observable If the measurement sensitivity matrix is invertible at any (continuous or discrete) time, then the system state can be uniquely determined (by inverting it) as $x = H^{-1}z$. In this case, the system model is considered to be *completely observable* at that time. However, the system can still be *observable over a time interval* even if H is not invertible at *any* time. In the latter case, the unique solution for the system state can be defined by using the least-squares methods of Chapter 1, including those of Sections 1.2.2 and 1.2.3 These use the so-called *Gramian matrix* to characterize whether or not a vector variable is determinable from a given linear model. When applied to the problem of the determinacy of the state of a linear dynamic system, the Gramian matrix is called the *observability matrix* of the given system model.

The observability matrix for dynamic system models in continuous time has the form

$$\mathcal{O}(H, F, t_0, t_f) = \int_{t_0}^{t_f} \Phi^{\mathrm{T}}(t) H^{\mathrm{T}}(t) H(t) \Phi(t)\, dt \tag{2.35}$$

for a linear dynamic system with fundamental solution matrix $\Phi(t)$ and measurement sensitivity matrix $H(t)$, defined over the continuous-time interval $t_0 \le t \le t_f$. Note that this depends on the interval over which the inputs and outputs are observed but not on the inputs and outputs per se. In fact, the observability matrix of a dynamic system model *does not* depend on the inputs u, the input coupling matrix C, or the input/output coupling matrix D—even though the outputs and the state vector depend on them. Because the fundamental solution matrix Φ depends only on the dynamic coefficient matrix F, the observability matrix depends *only* on H and F.

The observability matrix of a linear dynamic system model over a discrete-time interval $t_0 \le t \le t_{k_f}$ has the general form

$$\mathcal{O}(H_k, \Phi_k, 0 \le k \le k_f) = \left\{ \sum_{k=1}^{k_f} \left[\prod_{i=0}^{k-1} \Phi_{k-i} \right]^{\mathrm{T}} H_k^{\mathrm{T}} H_k \left[\prod_{i=0}^{k-1} \Phi_{k-i} \right] \right\}, \tag{2.36}$$

where H_k is the observability matrix at time t_k and Φ_k is the STM from time t_k to time t_{k+1} for $0 \le k \le k_f$. Therefore, the observability of discrete-time system models depends only on the values of H_k and Φ_k over this interval. As in the continuous-time case, observability does not depend on the system inputs.

The derivations of these formulas are left as exercises for the reader.

2.5.1 Observability of Time-Invariant Systems

The formulas defining observability are simpler when the dynamic coefficient matrices or state transition matrices of the dynamic system model are time invariant. In that case, observability can be characterized by the rank of the matrices

$$M = \begin{bmatrix} H^{\mathrm{T}} & \Phi^{\mathrm{T}} H^{\mathrm{T}} & (\Phi^{\mathrm{T}})^2 H^{\mathrm{T}} & \dots & (\Phi^{\mathrm{T}})^{n-1} H^{\mathrm{T}} \end{bmatrix} \tag{2.37}$$

for discrete-time systems and

$$M = \begin{bmatrix} H^{\mathrm{T}} & F^{\mathrm{T}} H^{\mathrm{T}} & (F^{\mathrm{T}})^2 H^{\mathrm{T}} & \dots & (F^{\mathrm{T}})^{n-1} H^{\mathrm{T}} \end{bmatrix} \tag{2.38}$$

for continuous-time systems. The systems are observable if these have rank n, the dimension of the system state vector. The first of these matrices can be obtained by representing the *initial state* of the linear dynamic system as a function of the system inputs and outputs. The initial state can then be shown to be uniquely determinable if and only if the rank condition is met. The derivation of the latter matrix is not as straightforward. Ogata [205] presents a derivation obtained by using properties of the characteristic polynomial of F.

2.5.1.1 Practicality of the Formal Definition of Observability Singularity of the observability matrix is a concise mathematical characterization of observability. This can be too fine a distinction for practical application—especially in finite-precision arithmetic—because arbitrarily small changes in the elements of a singular matrix can render it nonsingular. The following practical considerations should be kept in mind when applying the formal definition of observability:

- It is important to remember that the model is only an approximation to a real system, and we are primarily interested in the properties of the real system, not the model. Differences between the real system and the model are called *model truncation errors*. The art of system modeling depends on knowing where to truncate, but there will almost surely be some truncation error in any model.
- Computation of the observability matrix is subject to model truncation errors and *roundoff errors*, which could make the difference between singularity and nonsingularity of the result. Even if the computed observability matrix is *close* to being singular, it is cause for concern. One should consider a system as *poorly observable* if its observability matrix is close to being singular. For that purpose, one can use the *singular-value decomposition* or the *condition number* of the observability matrix to define a more *quantitative* measure of unobservability. The reciprocal of its condition number measures how close the system is to being unobservable.
- Real systems tend to have some unpredictability in their behavior due to unknown or neglected exogenous inputs. Although such effects cannot be modeled deterministically, they are not always negligible. Furthermore, the

process of measuring the outputs with physical sensors introduces some sensor noise, which will cause errors in the estimated state. It would be better to have a quantitative characterization of observability that takes these types of uncertainties into account. An approach to these issues (pursued in Chapter 4) uses a *statistical* characterization of observability, based on a statistical model of the uncertainties in the measured system outputs and the system dynamics. The degree of uncertainty in the estimated values of the system states can be characterized by an *information matrix*, which is a statistical generalization of the observability matrix.

Example 2.9 (Observability of $\dot{x}_1 = x_2$, $\dot{x}_2 = u(t)$ from $z = x_1$) Consider the following continuous system:

$$\dot{x}(t) = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t),$$

$$z(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} x(t).$$

The observability matrix, using Equation 2.38, is

$$M = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \text{ rank of } M = 2.$$

Here, M has rank equal to the dimension of $x(t)$. Therefore, the system is observable.

Example 2.10 (Observability for $z = x_2$) Consider the following continuous system:

$$\dot{x}(t) = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t),$$

$$z(t) = \begin{bmatrix} 0 & 1 \end{bmatrix} x(t).$$

The observability matrix, using Equation 2.38, is

$$M = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}, \text{ rank of } M = 1.$$

Here, M has rank less than the dimension of $x(t)$. Therefore, the system is not observable.

Example 2.11 (Unobservable Discrete System) Consider the following discrete system:

$$x_k = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix} x_{k-1} + \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} u_{k-1},$$

$$z_k = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} x_k.$$

The observability matrix, using Equation 2.37, is

$$M = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \text{ rank of } M = 2.$$

The rank is less than the dimension of x_k. Therefore, the system is not observable.

Example 2.12 (Observable Discrete System) Consider the following discrete system:

$$x_k = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} x_{k-1} + \begin{bmatrix} 2 \\ 1 \end{bmatrix} u_{k-1},$$

$$z_k = \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} x_k.$$

The observability matrix, using Equation 2.37, is

$$M = \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix}, \text{ rank of } M = 2.$$

The system is observable.

2.5.2 Controllability of Time-Invariant Linear Systems

2.5.2.1 Controllability in Continuous Time The concept of observability in estimation theory has algebraic relationships to the concept of *controllability* in control theory. These concepts and their relationships were discovered by R. E. Kalman as what he called the *duality* and *separability* of the estimation and control problems for linear dynamic systems. Kalman's[8] dual concepts are presented here and in the next subsection, although they are not issues for the estimation problem.

[8]The dual relationships between estimation and control given here are those originally defined by Kalman. These concepts have been refined and extended by later investigators to include concepts of *reachability* and *reconstructibility* as well. The interested reader is referred to the more recent textbooks on modern control theory for further exposition of these other "-ilities."

A dynamic system defined on the finite interval $t_0 \le t \le t_f$ by the linear model

$$\dot{x}(t) = Fx(t) + Cu(t), \qquad z(t) = Hx(t) + Du(t) \tag{2.39}$$

and with initial state vector $x(t_0)$ is said to be *controllable* at time $t = t_0$ if, for any desired final state $x(t_f)$, there exists a piecewise continuous input function $u(t)$ that drives to state $x(t_f)$. If every initial state of the system is controllable in some finite time interval, then the *system* is said to be controllable.

The system given in Equation 2.39 is controllable if and only if matrix S has n linearly independent columns,

$$S = [C \quad FC \quad F^2C \quad \cdots \quad F^{n-1}C]. \tag{2.40}$$

2.5.2.2 Controllability in Discrete Time Consider the time-invariant system model given by the equations

$$x_k = \Phi x_{k-1} + \Gamma u_{k-1}, \tag{2.41}$$

$$z_k = Hx_k + Du_k. \tag{2.42}$$

This system model is considered controllable[9] if there exists a set of control signals u_k defined over the discrete interval $0 \le k \le N$ that bring the system from an initial state x_0 to a given final state x_N in N sampling instants, where N is a finite positive integer. This condition can be shown to be equivalent to the matrix

$$S = [\Gamma \quad \Phi\Gamma \quad \Phi^2\Gamma \quad \cdots \quad \Phi^{N-1}\Gamma] \tag{2.43}$$

having rank n.

Example 2.13 (Controllable System) Determine the controllability of Example 2.8. The controllability matrix, using Equation 2.40, is

$$S = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad \text{rank of } S = 2.$$

Here, S has rank equal to the dimension of $x(t)$. Therefore, the system is controllable.

Example 2.14 (Uncontrollable System) Determine the controllability of Example 2.11. The controllability matrix, using Equation 2.43, is

$$S = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 2 & 0 \end{bmatrix}, \quad \text{rank of } S = 2.$$

The system is not controllable.

[9]This condition is also called *reachability*, with controllability restricted to $x_N = 0$.

2.6 SUMMARY

2.6.1 Systems and Processes

A *system* is a collection of interrelated objects treated as a whole for the purpose of modeling its behavior. It is called *dynamic* if attributes of interest are changing with time. A *process* is the evolution over time of a system.

2.6.2 Continuous and Discrete Time

Although it is sometimes convenient to model time as a continuum, it is often more practical to consider it as taking on discrete values. (Most clocks, for example, advance in discrete time steps.)

2.6.3 State Variables and Vectors

The *state* of a dynamic system at a given instant of time is characterized by the instantaneous values of its attributes of interest. For the problems of interest in this book, the attributes of interest can be characterized by real numbers, such as the electric potentials, temperatures, or positions of its component parts—in appropriate units. A *state variable* of a system is the associated real number. The *state vector* of a system has state variables as its component elements. The system is considered *closed* if the future state of the system for all time is uniquely determined by its current state. For example, neglecting the gravity fields from other massive bodies in the universe, the solar system could be considered as a closed system. If a dynamic system is not closed, then the exogenous causes are called *inputs* to the system. This state vector of a system must be *complete* in the sense that the future state of the system is uniquely determined by its current state *and its future inputs*.[10] In order to obtain a complete state vector for a system, one can extend the state variable components to include derivatives of other state variables. This allows one to use velocity (the derivative of position) or acceleration (the derivative of velocity) as state variables, for example.

2.6.4 State-Space Models for Dynamic Systems

In order that the future state of a system may be determinable from its current state and future inputs, the dynamic behavior of each state variable of the system must be a known function of the instantaneous values of other state variables and the system inputs. In the canonical example of our solar system, for instance, the acceleration of each body is a known function of the relative positions of the other bodies. The *state-space model* for a dynamic system represents these functional dependencies

[10]This concept in the state-space approach will be generalized in the next chapter to the state of knowledge about a system, characterized by the *probability distribution* of its state variables. That is, the *future* probability distribution of the system state variables will be uniquely determined by their *present* probability distribution and the probability distributions of *future inputs*.

in terms of first-order *differential equations* (in continuous time) or *difference equations* (in discrete time). The differential or difference equations representing the behavior of a dynamic system are called its *state equations*. If these can be represented by *linear* functions, then it is called a *linear dynamic system*.

2.6.5 Linear Dynamic System Models

The model for a linear dynamic system in continuous time can be expressed in general form as a first-order vector differential equation

$$\frac{d}{dt}x(t) = F(t)x(t) + C(t)u(t),$$

where $x(t)$ is the n-dimensional *system state vector* at time t, $F(t)$ is its $n \times n$ *dynamic coefficient matrix,* $u(t)$ is the r-dimensional *system input vector*, and $C(t)$ is the $n \times r$ *input coupling matrix*. The corresponding model for a linear dynamic system in discrete time can be expressed in the general form

$$x_k = \Phi_{k-1}x_{k-1} + \Gamma_{k-1}u_{k-1},$$

where x_{k-1} is the n-dimensional system state vector at time t_{k-1}, x_k is its value at time $t_k > t_{k-1}$, Φ_{k-1} is the $n \times n$ *state transition matrix* for the system at time t_k, u_k is the input vector to the system a time t_k, and Γ_k is the corresponding input coupling matrix.

2.6.6 Time-Varying and Time-Invariant Dynamic Systems

If F and C (or Φ and Γ) do not depend upon t (or k), then the continuous (or discrete) model is called *time invariant*. Otherwise, the model is *time-varying*.

2.6.7 Homogeneous Systems and Fundamental Solution Matrices

The equation

$$\frac{d}{dt}x(t) = F(t)x(t)$$

is called the *homogeneous part* of the model equation

$$\frac{d}{dt}x(t) = F(t)x(t) + C(t)u(t).$$

A solution $\Phi(t)$ to the corresponding $n \times n$ *matrix equation*

$$\frac{d}{dt}\Phi(t) = F(t)\Phi(t)$$

on an interval starting at time $t = t_0$ and with initial condition

$$\Phi(t_0) = I \quad \text{(the identity matrix)}$$

is called a *fundamental solution matrix* to the homogeneous equation on that interval. It has the property that, if the elements of $F(t)$ are bounded, then $\Phi(t)$ cannot become singular on a finite interval. Furthermore, for any initial value $x(t_0)$,

$$x(t) = \Phi(t)x(t_0)$$

is the solution to the corresponding homogeneous equation.

2.6.8 Fundamental Solution Matrices and State Transition Matrices

For a *homogeneous* system, the state transition matrix Φ_{k-1} from time t_{k-1} to time t_k can be expressed in terms of the fundamental solution $\Phi(t)$ as

$$\Phi_{k-1} = \Phi(t_k)\Phi^{-1}(t_{k-1})$$

for times $t_k > t_{k-1} \geq t_0$.

2.6.9 Transforming Continuous-Time Models into Discrete-Time Models

The model for a dynamic system in continuous time can be transformed into a model in discrete time using the above formula for the state transition matrix and the following formula for the equivalent discrete-time inputs:

$$\Gamma u_{k-1} = \Phi(t_k) \int_{t_{k-1}}^{t_k} \Phi^{-1}(\tau)C(\tau)u(\tau)\, d\tau.$$

2.6.10 Linear System Output Models and Observability

An *output* of a dynamic system is something we can measure directly, such as directions of the lines of sight to the planets (viewing conditions permitting) or the temperature at a thermocouple. A dynamic system model is said to be *observable* from a given set of outputs if it is feasible to determine the state of the system from those outputs. If the dependence of an output z on the system state x is linear, it can be expressed in the form

$$z = Hx,$$

where H is called the *measurement sensitivity matrix*. It can be a function of continuous time $[H(t)]$ or discrete time (H_k). Observability can be characterized by the rank

of an *observability matrix* associated with a given system model. The observability matrix is defined as

$$\mathcal{O} = \begin{cases} \int_{t_0}^{t} \Phi^{\mathrm{T}}(\tau) H^{\mathrm{T}}(\tau) H(\tau) \Phi(\tau)\, d\tau & \text{for continuous-time models,} \\ \sum_{i=0}^{m} \left[\left(\prod_{k=0}^{i-1} \Phi_k^{\mathrm{T}} \right) H_i^{\mathrm{T}} H_i \left(\prod_{k=0}^{i-1} \Phi_k^{\mathrm{T}} \right)^{\mathrm{T}} \right] & \text{for discrete-time models.} \end{cases}$$

The system is observable if and only if its observability matrix has full rank (n) for some integer $m \geq 0$ or time $t > t_0$. (The test for observability can be simplified for time-invariant systems.) Note that the determination of observability depends on the (continuous or discrete) interval over which the observability matrix is determined.

2.6.11 Reliable Numerical Approximation of Matrix Exponential

The closed-form solution of a system of first-order differential equations with constant coefficients can be expressed symbolically in terms of the exponential function of a matrix, but the problem of numerical approximation of the exponential function of a matrix is notoriously ill-conditioned. (See Section B.4.4.2.)

PROBLEMS

2.1 What is a state vector model for the linear dynamic system $\frac{dy(t)}{dt} = u(t)$, expressed in terms of y? (Assume the companion form of the dynamic coefficient matrix.)

2.2 What is the companion matrix for the nth-order differential equation $(d/dt)^n y(t) = 0$? What are its dimensions?

2.3 What is the companion matrix of the above problem when $n = 1$? When $n = 2$?

2.4 What is the fundamental solution matrix of Exercise 2.2 when $n = 1$? When $n = 2$?

2.5 What is the state transition matrix of the above problem when $n = 1$? When $n = 2$?

2.6 Find the fundamental solution matrix $\Phi(t)$ for the system

$$\frac{d}{dt}\begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ -1 & -2 \end{bmatrix}\begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

and the solution $x(t)$ for the initial conditions

$$x_1(0) = 1 \quad \text{and} \quad x_2(0) = 2.$$

2.7 Find the total solution and the state transition matrix for the system

$$\frac{d}{dt}\begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}\begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 5 \\ 1 \end{bmatrix}$$

with initial conditions $x_1(0) = 1$ and $x_2(0) = 2$.

2.8 The reverse problem: from a discrete-time model to a continuous-time model. For the discrete-time dynamic system model

$$x_k = \begin{bmatrix} 0 & 1 \\ -1 & 2 \end{bmatrix} x_{k-1} + \begin{bmatrix} 0 \\ 1 \end{bmatrix},$$

find the state transition matrix for continuous time and the solution for the continuous-time system with initial conditions

$$x(0) = \begin{bmatrix} 1 \\ 2 \end{bmatrix}.$$

Also find the total discrete-time solution.

2.9 Find conditions on c_1, c_2, h_1, h_2 such that the following system is completely observable and controllable:

$$\frac{d}{dt}\begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} u(t),$$

$$z(t) = \begin{bmatrix} h_1 & h_2 \end{bmatrix}\begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}.$$

2.10 Determine the controllability and observability of the following dynamic system model:

$$\frac{d}{dt}\begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}\begin{bmatrix} u_1 \\ u_2 \end{bmatrix},$$

$$z(t) = \begin{bmatrix} 0 & 1 \end{bmatrix}\begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}.$$

2.11 Derive the state transition matrix of the time-varying system

$$\dot{x}(t) = \begin{bmatrix} t & 0 \\ 0 & t \end{bmatrix} x(t).$$

2.12 Find the state transition matrix for

$$F = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

2.13 For the system of three first-order differential equations

$$\dot{x}_1 = x_2, \quad \dot{x}_2 = x_3, \quad \dot{x}_3 = 0$$

(a) What is the companion matrix F?

(b) What is the fundamental solution matrix $\Phi(t)$ such that $(d/dt)\Phi(t) = F\Phi(t)$ and $\Phi(0) = I$?

2.14 Show that the matrix exponential of an antisymmetric matrix is an orthogonal matrix.

2.15 Derive the formula of Equation 2.35 for the observability matrix of a linear dynamic system model in continuous time. (Hint: Use the approach of Example 1.2 for estimating the initial state of a system and Equation 2.20 for the state of a system as a linear function of its initial state and its inputs.)

2.16 Derive the formula of Equation 2.36 for the observability matrix of a dynamic system in discrete time. (*Hint*: Use the method of least squares of Example 1.1 for estimating the initial state of a system, and compare the resulting Gramian matrix to the observability matrix of Equation 2.36.)

3

RANDOM PROCESSES AND STOCHASTIC SYSTEMS

A completely satisfactory definition of random sequence is yet to be discovered.
—G. James and R. C. James, *Mathematics Dictionary*
(D. Van Nostrand Co., Princeton, New Jersey, 1959)

3.1 CHAPTER FOCUS

Deterministic Mechanics The previous chapter described models for dynamic systems with manageable numbers of moving parts. These are models for *deterministic mechanics*, in which the state of every component of the system is represented and propagated explicitly.

Statistical Mechanics Another approach has been developed for extremely large dynamic systems, such as the ensemble of gas molecules in a reaction chamber. The state-space approach for such large systems would be impractical. Consequently, this other approach focuses on the ensemble *statistical* properties of the system and treats the underlying dynamics as a *random process*. The results are models for *statistical mechanics*, in which only the ensemble statistical properties of the system are represented and propagated explicitly.

Stochastic Systems In this chapter, some of the basic notions and mathematical models of statistical and deterministic mechanics are combined into a *stochastic*

Kalman Filtering: Theory and Practice using MATLAB®, Third Edition. By Mohinder S. Grewal and Angus P. Andrews

system model, which represents the *state of knowledge* about a dynamic system. These models represent *what we know* about a dynamic system, including a quantitative model for our *uncertainty* about what we know.

Random Processes and Sequences Stochastic system models are used to define random processes in continuous time and in discrete time (also called *random sequences*).

In the next chapter, methods will be derived for modifying that state of knowledge, based on noisy measurements (i.e., sensor outputs) related to the state of the dynamic system.

3.1.1 Discovery and Modeling of Random Processes (RPs)

Modeling Financial Markets We mentioned in Chapter 1 the early history of analysis and modeling of unpredictable events in gambling. Financial markets—which some prefer to ordinary gambling—came under similar analysis in the nineteenth century, when Carl Friedrich Gauss (1777–1855) did quite well in managing his own investments as well as those of widows of professors at the University of Göttingen. The Danish astronomer-turned-actuary Thorvald Nicolai Thiele (1838–1910) did some seminal work on modeling of RPs and sequences, and the French mathematician Louis Bachelier (1870–1946) developed models for prices on the Paris Bourse, the French stock exchange. These developments in stochastic economics received scant attention among other scientists and engineers until quite recently, but the underappreciated synergy between mathematical economics and mathematical engineering has continued to this day. There is no Nobel Prize in either mathematics or engineering, but ever since the Bank of Sweden established the Nobel Prize in Economics in 1969, many of its recipients have been mathematical economists.

Brownian Motion The British botanist Robert Brown (1773–1858) reported in 1827 a phenomenon he had observed while studying pollen grains of the herb *Clarkia pulchella* suspended in water and similar observations by earlier investigators. The particles appeared to move about erratically, as though propelled by some unknown force. This phenomenon came to be called *Brownian movement* or *Brownian motion*. It was studied extensively—both empirically and theoretically—by many eminent scientists (including Albert Einstein [80]) for much of the twentieth century. Empirical studies demonstrated that no biological forces were involved and eventually established that individual collisions with molecules of the surrounding fluid were causing the motion observed. The empirical results quantified how some statistical properties of the random motion were influenced by such physical properties as the size and mass of the particles and the temperature and viscosity of the surrounding fluid.

The Langevin Equation Mathematical models for RPs were derived in terms of what has come to be called *stochastic differential equations*. The French scientist

Paul Langevin[1] (1872–1946) modeled the velocity v of a particle in Brownian motion in terms of a differential equation [162] of the form

$$\frac{dv}{dt} = -\beta v + a(t), \tag{3.1}$$

where β is a damping coefficient (due to the viscosity of the suspending medium) and $a(t)$ is called a *random force*. Equation (3.1) is now called the *Langevin equation*.

Stochastic Differential Equations The random forcing function $a(t)$ of the Langevin equation (3.1) has been idealized in three ways from the physically motivated example of Brownian motion:

1. The velocity changes imparted to the particle have been assumed to be statistically independent from one collision to another.
2. The mean velocity change from independent collisions is assumed to be zero.
3. The effective time between collisions has been allowed to shrink to zero, with the magnitude of the imparted velocity change shrinking accordingly.

Stochastic Calculus This new process model for $a(t)$ transcends the ordinary (Riemann) calculus, because the resulting "white-noise" process $a(t)$ is not integrable in the ordinary calculus of Georg Friedrich Bernhard Riemann (1826–1866). At about the time that the Kalman filter was introduced, however, a special calculus was developed by Kiyosi Itô (called the *Itô calculus* or the *stochastic calculus*) to handle such functions. Equivalent derivations were done by the Russian mathematician Ruslan L. Stratonovich (1930–1997) [262] and others, and the stochastic calculus is now commonly used in modeling of RPs.

White Noise and Wiener Processes Another mathematical characterization of white noise was provided by Norbert Weiner, using his generalized harmonic analysis. Wiener preferred to focus on the mathematical properties of $v(t)$ in Equation (3.1) with $\beta = 0$, a process now called a *Wiener process*.

3.1.2 Main Points to Be Covered

The theory of RPs and stochastic systems represents the evolution over time of the uncertainty of our knowledge about physical systems. This representation includes the effects of any *measurements* (or *observations*) that we make of the physical process and the effects of uncertainties about the measurement processes and dynamic processes involved. The uncertainties in the measurement and dynamic processes are modeled by RPs and stochastic systems.

[1]Lengevin is also famous for pioneering the development of sonar in World War I.

Properties of uncertain dynamic systems are characterized by statistical parameters such as *means, correlations*, and *covariances*. By using only these numerical parameters, one can obtain finite representations of some probability distributions, which is important for implementing the solution on digital computers. This representation depends upon such statistical properties as orthogonality, stationarity, ergodicity, and Markovianness of the RPs involved and the Gaussianity of probability distributions. Gaussian, Markov, and uncorrelated (white-noise) processes will be used extensively in the following chapters. The autocorrelation functions and power spectral densities (PSDs) of such processes are also used. These are important in the development of frequency-domain and time-domain models. The time-domain models may be either continuous or discrete.

Shaping filters (continuous and discrete) are developed as models for many applications encountered in practice. These include random constants, random walks and ramps, sinusoidally correlated processes, and exponentially correlated processes. We derive the linear covariance equations for continuous and discrete systems to be used in Chapter 4. The *orthogonality principle* is developed and explained with scalar examples. This principle will be used in Chapter 4 to derive the Kalman filter equations.

3.1.3 Topics Not Covered

It is assumed that the reader is already familiar with the mathematical foundations of probability theory, as covered by Grinstead and Snell [107], Papoulis [208], or Billingsley [33], for example. The treatment of these concepts in this chapter is heuristic and very brief. The reader is referred to textbooks of this type for more detailed background material.

The Itô calculus for the integration of otherwise nonintegrable functions (white noise, in particular) is not defined, although it is used. The interested reader is referred to books on the mathematics of stochastic differential equations (e.g., those by Allen [4], Arnold [16], Baras and Mirelli [21], Itô and McKean [115], Oksendal [206], Sobczyk [251], or Stratonovich [262]).

3.2 PROBABILITY AND RANDOM VARIABLES (RVs)

The relationships between unknown physical processes, probability spaces, and RVs are illustrated in Figure 3.1. The behavior of the physical processes is investigated by what is called a *statistical experiment*, which helps to define a model for the physical process as a probability space. Strictly speaking, this is not a model for the physical process itself, but rather a model of our own understanding of the physical process. It defines what might be called our *state of knowledge* about the physical process, which is essentially a model for our *uncertainty* about the physical process.

An RV represents a *numerical attribute* of the state of the physical process. In the following subsections, these concepts are illustrated by using the numerical score from tossing dice as an example of a random variable.

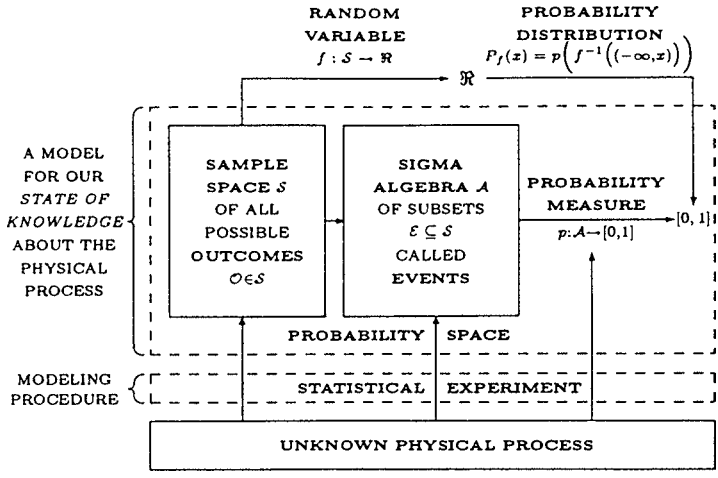


Figure 3.1 Conceptual model for a random variable.

Example 3.1 (Score from Tossing a Die) A die (plural is dice) is a cube with its six faces marked by patterns of one to six dots. It is thrown onto a flat surface such that it tumbles about and comes to rest with one of these faces on top. This can be considered an unknown process in the sense that which face will wind up on top is not reliably predictable before the toss. The tossing of a die in this manner is an example of a statistical experiment for defining a statistical model for the process. Each toss of the die can result in but one outcome, corresponding to which one of the six faces of the die is on top when it comes to rest. Let us label these outcomes $\mathcal{O}_a$, $\mathcal{O}_b$, $\mathcal{O}_c$, $\mathcal{O}_d$, $\mathcal{O}_e$, $\mathcal{O}_f$. The set of all possible outcomes of a statistical experiment is called a *sample space*. The sample space for the statistical experiment with one die is the set $\mathcal{S} = \{\mathcal{O}_a, \mathcal{O}_b, \mathcal{O}_c, \mathcal{O}_d, \mathcal{O}_e, \mathcal{O}_f\}$.

A Random Variable Assigns Real Numbers to Outcomes There is an integral number of dots on each face of the die. This defines a "dot function" $d : \mathcal{S} \to \mathcal{R}$ on the sample space $\mathcal{S}$, where $d(\mathcal{O})$ is the number of dots showing for the outcome $\mathcal{O}$ of the statistical experiment. Assign the values

$$d(\mathcal{O}_a) = 1, \quad d(\mathcal{O}_c) = 3, \quad d(\mathcal{O}_e) = 5,$$
$$d(\mathcal{O}_b) = 2, \quad d(\mathcal{O}_d) = 4, \quad d(\mathcal{O}_f) = 6.$$

This *function* is an example of an RV. The useful statistical properties of this RV will depend upon the probability space defined by statistical experiments with the die.

Events and Sigma Algebras The statistical properties of the RV d depend on the probabilities of sets of outcomes (called *events*) forming what is called a

sigma algebra[2] of subsets of the sample space $\mathcal{S}$. Any collection of events that includes the sample space itself, the *empty set* (the set with no elements), and the *set unions* and *set complements* of all its members is called a *sigma algebra* over the sample space. The set of *all subsets* of $\mathcal{S}$ is a sigma algebra with $2^6 = 64$ events.

The Probability Space for a Fair Die Adie is considered fair if, in a large number of tosses, all outcomes tend to occur with equal frequency. The *relative frequency* of any outcome is defined as the ratio of the number of occurrences of that outcome to the number of occurrences of all outcomes. Relative frequencies of outcomes of a statistical experiment are called *probabilities*. Note that, by this definition, the sum of the probabilities of all outcomes will always be equal to 1. This defines a probability $p(\mathcal{E})$ for every event $\mathcal{E}$ (a set of outcomes) equal to

$$p(\mathcal{E}) = \frac{\#(\mathcal{E})}{\#(\mathcal{S})},$$

where $\#(\mathcal{E})$ is the *cardinality* of $\mathcal{E}$, equal to the *number* of outcomes $\mathcal{O} \in \mathcal{E}$. Note that this assigns probability 0 to the empty set and probability 1 to the sample space.

The *probability distribution* of the RV d is a nondecreasing function $P_d(x)$ defined for every real number x as the probability of the event for which the score is less than x. It has the formal definition

$$\begin{aligned} P_d(x) &\stackrel{\text{def}}{=} p(d^{-1}(-\infty, x)), \\ d^{-1}((-\infty, x)) &\stackrel{\text{def}}{=} \{\mathcal{O} |\, d(\mathcal{O}) \leq x\} \end{aligned}.$$

For every real value of x, the set $\{\mathcal{O}|d\,(\mathcal{O}) < x\}$ is an *event*. For example,

$$\begin{aligned} P_d(1) &= p(d^{-1}((-\infty,1))) \\ &= p(\{\mathcal{O} |\, d(\mathcal{O}) < 1\}) \\ &= p(\{\}) \quad \text{(the empty set)} \\ &= 0, \\ P_d(1.0\cdots01) &= p(d^{-1}((-\infty,1.0\cdots01))) \\ &= p(\{\mathcal{O} |\, d(\mathcal{O}) < 1.0\cdots01\}) \\ &= p(\{\mathcal{O}_a\}) = \frac{1}{6}, \\ &\vdots \\ P_d(6.0\cdots01) &= p(\mathcal{S}) = 1, \end{aligned}$$

[2]Such a collection of subsets $\mathcal{E}_i$ of a set $\mathcal{S}$ is called an *algebra* because it is a Boolean algebra with respect to the operations of set union ($\mathcal{E}_1 \cup \mathcal{E}_2$), set intersection ($\mathcal{E}_1 \cap \mathcal{E}_2$), and set complement ($\mathcal{S}|\mathcal{E}$)—corresponding to the logical operations *or, and*, and *not*, respectively. The "sigma" refers to the summation symbol Σ, which is used for defining the additive properties of the associated probability measure. However, the lowercase symbol σ is used for abbreviating *sigma algebra* to *σ-algebra.*

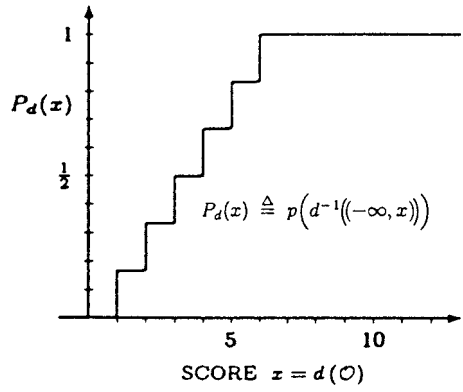


Figure 3.2 Probability distribution of scores from a fair die.

as plotted in Figure 3.2. Note that P_d *is not a continuous function* in this particular example.

3.2.1 Probability Distributions and Densities

RVs f are required to have the property that, for every real a and b such that $-\infty \leq a \leq b \leq +\infty$, the outcomes $\mathcal{O}$ such that $a < f(\mathcal{O}) < b$ are an event $\mathcal{E} \in \mathcal{A}$. This property is needed for defining the *probability distribution function* P_f of f as

$$P_f(x) \stackrel{\text{def}}{=} p(f^{-1}((-\infty, x))), \tag{3.2}$$

$$f^{-1}((-\infty, x)) \stackrel{\text{def}}{=} \{\mathcal{O} \in \mathcal{S} | f(\mathcal{O}) \leq x\}. \tag{3.3}$$

The probability distribution function may not be a differentiable function. However, if it *is* differentiable, then its derivative

$$p_f(x) = \frac{d}{dx} P_f(x) \tag{3.4}$$

is called the *probability density function* of the RV, f, and the differential

$$p_f(x)\, dx = dP_f(x) \tag{3.5}$$

is the *probability measure* of f defined on a sigma algebra containing the open intervals (called the *Borel*[3] *algebra* over $\mathcal{R}$).

[3]Named for the French mathematician Félix Borel (1871–1956).

A *vector-valued* RV is a vector with RVs as its components. An analogous derivation applies to vector-valued RVs, for which the analogous probability measures are defined on the Borel algebras over $\mathcal{R}^n$.

3.2.2 Gaussian Probability Density Functions

The probability distribution of the average score from tossing n dice (i.e., the total number of dots divided by the number of dice) tends toward a particular type of distribution as $n \to \infty$ called a *Gaussian distribution*.[4] It is the limit of many such distributions, and it is common to many models for random phenomena. It is commonly used in stochastic system models for the distributions of random variables.

Univariate Gaussian Probability Distributions The notation $\mathcal{N}(\bar{x}, \sigma^2)$ is used to denote a probability distribution with density function

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{1}{2}\frac{(x-\bar{x})^2}{\sigma^2}\right], \tag{3.6}$$

where

$$\bar{x} = \mathrm{E}\langle x \rangle \tag{3.7}$$

is the *mean* of the distribution (a term that will be defined in Section 3.4.2) and σ^2 is its *variance* (also defined in Section 3.4.2). The $\mathcal{N}$ stands for *normal*, another name for the Gaussian distribution. Because so many other things are called *normal* in mathematics, it is less confusing if we call it *Gaussian*.

Gaussian Expectation Operators and Generating Functions Because the Gaussian probability density function depends only on the difference $x - \bar{x}$, the expectation operator

$$\underset{x}{\mathrm{E}}\langle f(x) \rangle = \int_{-\infty}^{+\infty} f(x)p(x)\,dx \tag{3.8}$$

$$= \frac{1}{\sqrt{2\pi}\sigma} \int_{-\infty}^{+\infty} f(x)e^{-(x-\bar{x})^2/2\sigma^2}\,dx \tag{3.9}$$

$$= \frac{1}{\sqrt{2\pi}\sigma} \int_{-\infty}^{+\infty} f(x+\bar{x})e^{-x^2/2\sigma^2}\,dx \tag{3.10}$$

has the form of a convolution integral. This has important implications for problems in which it must be implemented numerically, because the convolution can be

[4]It is called the *Laplace distribution* in France. It has had many discoverers besides Gauss and Laplace, including the American mathematician Robert Adrian (1775–1843). The physicist Gabriel Lippman (1845–1921) is credited with the observation that "mathematicians think it [the normal distribution] is a law of nature and physicists are convinced that it is a mathematical theorem."

implemented more efficiently as a fast Fourier transform of f, followed by a pointwise product of its transform with the Fourier transform of p, followed by an inverse fast Fourier transform of the result. One does not need to take the numerical Fourier transform of p, because its Fourier transform can be expressed analytically in closed form. Recall that the Fourier transform of p is called its *generating function*.

Fourier Transforms of Gaussian Probability Density Functions Fourier transforms of Gaussian probability density functions are also (possibly scaled) Gaussian density functions:

$$p(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} p(x) e^{i\omega x}\, dx \tag{3.11}$$

$$= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \frac{e^{-x^2/2\sigma^2}}{\sqrt{2\pi\sigma}} e^{i\omega x}\, dx \tag{3.12}$$

$$= \frac{\sigma}{\sqrt{2\pi}} e^{(-1/2)\omega^2\sigma^2}, \tag{3.13}$$

a Gaussian density function with variance σ^{-2}. Here we have used a probability-preserving form of the Fourier transform, defined with the factor of $1/\sqrt{2\pi}$ in front of the integral. If other forms of the Fourier transform are used, the result is not a probability distribution but a scaled probability distribution.

3.2.2.1 Vector-Valued (Multivariate) Gaussian Distributions The formula for the n-dimensional Gaussian distribution $\mathcal{N}(\bar{x}, P)$, where the mean $\bar{x}$ is an n-vector and the covariance P is an $n \times n$ symmetric positive-definite matrix, is

$$p(x) = \frac{1}{\sqrt{(2\pi)^n \det P}} e^{(-1/2)(x-\bar{x})^{\mathrm{T}} P^{-1} (x-\bar{x})}. \tag{3.14}$$

The multivariate Gaussian generating function has the form

$$p(\omega) = \frac{1}{\sqrt{(2\pi)^n \det P^{-1}}} e^{(-1/2)\, \omega^{\mathrm{T}} P \omega}, \tag{3.15}$$

where ω is an n-vector. This is also a multivariate Gaussian probability distribution $\mathcal{N}(0, P^{-1})$ if the scaled form of the Fourier transform shown in Equation (3.11) is used.

3.2.3 Joint Probabilities and Conditional Probabilities

The *joint probability* of two events $\mathcal{E}_a$ and $\mathcal{E}_b$ is the probability of their *set intersection* $p(\mathcal{E}_a \cap \mathcal{E}_b)$, which is the probability that *both* events occur. The joint probability of *independent events* is the *product* of their probabilities.

The *conditional probability* of event $\mathcal{E}$, given that event $\mathcal{E}_c$ has occurred, is defined as the probability of $\mathcal{E}$ in the "conditioned" probability space with sample space $\mathcal{E}_c$. This is a probability space defined on the sigma algebra

$$\mathcal{A}|\mathcal{E}_c = \{\mathcal{E} \cap \mathcal{E}_c | \mathcal{E} \in \mathcal{A}\} \tag{3.16}$$

of the set intersections of all events $\mathcal{E} \in \mathcal{A}$ (the original sigma algebra) with the conditioning event $\mathcal{E}_c$. The probability measure on the conditioned sigma algebra $\mathcal{A}|\mathcal{E}_c$ is defined in terms of the joint probabilities in the original probability space by the rule

$$p(\mathcal{E}|\mathcal{E}_c) = \frac{p(\mathcal{E} \cap \mathcal{E}_c)}{p(\mathcal{E}_c)}, \tag{3.17}$$

where $p(\mathcal{E} \cap \mathcal{E}_c)$ is the joint probability of $\mathcal{E}$ and $\mathcal{E}_c$. Equation (3.17) is called *Bayes' rule*.[5]

Example 3.2 (Experiment with Two Dice) Consider a toss with two dice in which one die has come to rest before the other and just enough of its face is visible to show that it contains either four or five dots. The question is: What is the probability distribution of the score, given that information?

The Probability Space for Two Dice This example illustrates just how rapidly the sizes of probability spaces grow with the "problem size" (in this case, the number of dice). For a single die, the sample space has six outcomes and the sigma algebra has 64 events. For two dice, the sample space has 36 possible outcomes (6 independent outcomes for each of two dice) and $2^{36} = 68{,}719{,}476{,}736$ possible events. If each die is fair and their outcomes are independent, then all outcomes with two dice have probability $\left(\frac{1}{6}\right) \times \left(\frac{1}{6}\right) = \frac{1}{36}$ and the probability of any event is the number of outcomes in the event divided by 36 (the number of outcomes in the sample space). Using the same notation as in the previous (one-die) example, let the outcome from tossing a pair of dice be represented by an ordered pair (in parentheses) of the outcomes of the first and second die, respectively. Then the score $s((\mathcal{O}_i, \mathcal{O}_j)) = d(\mathcal{O}_i) + d(\mathcal{O}_j)$, where $\mathcal{O}_i$ represents the outcome of the first die and $\mathcal{O}_j$ represents the outcome of the second die. The corresponding probability distribution function of the score x for two dice is shown in Figure 3.3(*a*).

[5]Discovered by the English clergyman and mathematician Thomas Bayes (1702–1761). Conditioning on impossible events is not defined. Note that the conditional probability is based on the assumption that $\mathcal{E}_c$ *has occurred*. This would seem to imply that $\mathcal{E}_c$ is an event with *nonzero probability*, which one might expect from practical applications of Bayes' rule.

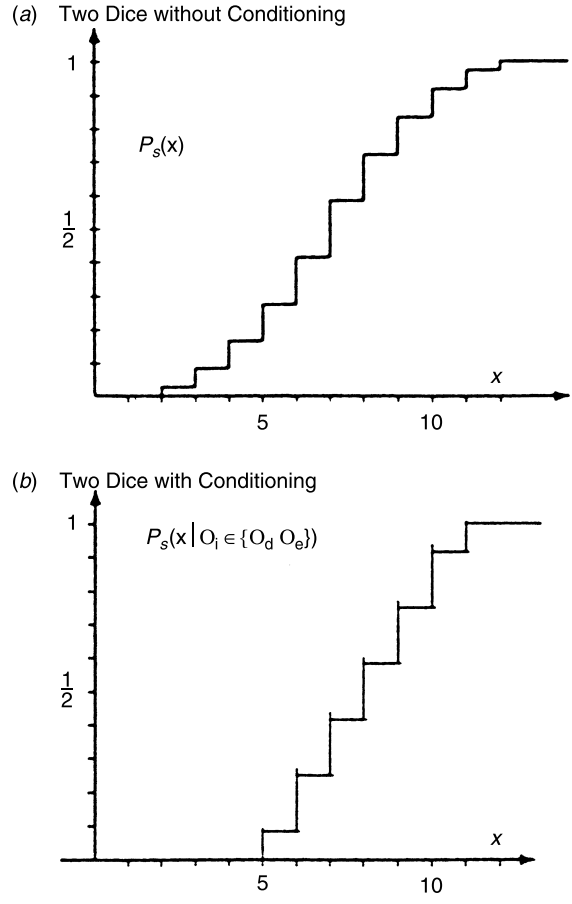


Figure 3.3 Probability distributions of dice scores.

The event corresponding to the condition that the first die have either four or five dots showing contains all outcomes in which $\mathcal{O}_i = \mathcal{O}_d$ *or* $\mathcal{O}_e$, which is the set

$$\mathcal{E}_c = \{(\mathcal{O}_d, \mathcal{O}_a), (\mathcal{O}_d, \mathcal{O}_b), (\mathcal{O}_d, \mathcal{O}_c), (\mathcal{O}_d, \mathcal{O}_d), (\mathcal{O}_d, \mathcal{O}_e), (\mathcal{O}_d, \mathcal{O}_f)$$
$$(\mathcal{O}_e, \mathcal{O}_a), (\mathcal{O}_e, \mathcal{O}_b), (\mathcal{O}_e, \mathcal{O}_c), (\mathcal{O}_e, \mathcal{O}_d), (\mathcal{O}_e, \mathcal{O}_e), (\mathcal{O}_e, \mathcal{O}_f)\}$$

of 12 outcomes. It has probability $p(\mathcal{E}_c) = \dfrac{12}{36} = \dfrac{1}{3}$.

By applying Bayes' rule, the conditional probabilities of all events corresponding to unique scores can be calculated as shown in Figure 3.4. The corresponding probability distribution function for two dice with this conditioning is shown in Figure 3.3(*b*).

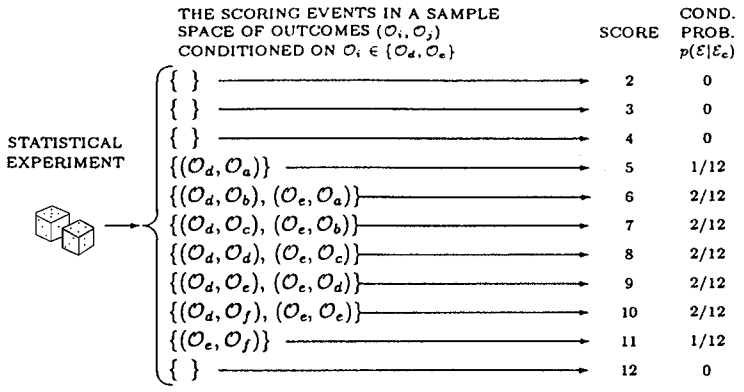


Figure 3.4 Conditional scoring probabilities for two dice.

3.3 STATISTICAL PROPERTIES OF RVs

3.3.1 Expected Values of RVs

Expected Values The symbol E is used as an operator on RVs. It is called the *expectancy, expected value*, or *average* operator, and the expression $\mathrm{E}_x\langle f(x)\rangle$ is used to denote the expected value of the function f applied to the ensemble of possible values of the RV x. The symbol under the E indicates the RV over which the expected value is to be evaluated. When the RV in question is obvious from the context, the symbol underneath the E will be eliminated. If the argument of the expectancy operator is also obvious from the context, the angle brackets can also be eliminated, using Ex instead of E$\langle x\rangle$, for example.

Moments The nth moment of a scalar RV x with probability density $p(x)$ is defined by the formula

$$\eta_n(x) \stackrel{\text{def}}{=} \mathop{\mathrm{E}}_{x}\langle x^n\rangle \stackrel{\text{def}}{=} \int_{\infty}^{\infty} x^n p(x)\,dx. \tag{3.18}$$

The nth *central moment* of x is defined as

$$\mu_n(x) \stackrel{\text{def}}{=} E\langle x - Ex\rangle^n \tag{3.19}$$

$$= \int_{-\infty}^{\infty} (x - Ex)^n p(x)\,dx. \tag{3.20}$$

The first moment of x is called its *mean*[6]:

$$\eta_1 = Ex = \int_{-\infty}^{\infty} xp(x)\,dx. \tag{3.21}$$

[6]Here, we restrict the order of the moment to the positive integers. Otherwise, the zeroth-order moment would always evaluate to 1.

In general, a function of several arguments such as *f(x, y, z)* has first moment

$$Ef(x,y,z) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y,z)p(z,y,z)\,dx\,dy\,dz. \tag{3.22}$$

Array Dimensions of Moments The first moment will be a scalar or a vector, depending on whether the function $f(x, y, z)$ is scalar or vector valued. Higher order moments have tensor-like properties, which we can characterize in terms of the number of subscripts used in defining them as data structures. Vectors are singly subscripted data structures. The higher order moments of vector-valued variates are successively higher order data structures. That is, the second moments of vector-valued RVs are matrices (doubly subscripted data structures), and the third-order moments will be triply subscripted data structures.

These definitions of a moment apply to discrete-valued RVs if we simply substitute summations in place of integrations in the definitions.

3.3.2 Functions of RVs

A function of an RV x is the operation of assigning to each value of x another value, for example y, according to rule or function. This is represented by

$$y = f(x), \tag{3.23}$$

where x and y are usually called *input* and *output*, respectively. The statistical properties of y in terms of x are, for example,

$$\begin{aligned} Ey &= \int_{-\infty}^{\infty} f(x)p(x)\,dx, \\ &\vdots \\ Ey^n &= \int_{-\infty}^{\infty} [f(x)]^n p(x)\,dx \end{aligned} \tag{3.24}$$

when y is scalar. For vector-valued functions y, similar expressions can be shown.

The probability density of y can be obtained from the density of x. If Equation (3.23) can be solved for x, yielding the unique solution

$$x = g(y), \tag{3.25}$$

then we have

$$p_y(y) = \frac{p_x(g(y))}{\dfrac{\partial f(x)}{\partial x}\Big|_{x\,=\,g(y)}}, \tag{3.26}$$

where $p_y(y)$ and $p_x(x)$ are the density functions of y and x, respectively. A function of two RVs, x,y, is the process of assigning to each pair of x,y another value, for example z, according to the same rule,

$$z = f(y, x), \tag{3.27}$$

and similarly functions of n RVs. When x and y in Equation (3.23) are n-dimensional vectors and if a unique solution for x in terms of y exists,

$$x = g(y). \tag{3.28}$$

Equation (3.26) becomes

$$p_y(y) = \frac{p_x[g(y)]}{|J|_{x=g(y)}}, \tag{3.29}$$

where the Jacobian $|J|$ is defined as the determinant of the array of partial derivatives $\partial f_i/\partial x_j$:

$$|J| = \det \begin{bmatrix} \dfrac{\partial f_1}{\partial x_1} & \dfrac{\partial f_1}{\partial x_2} & \cdots & \dfrac{\partial f_1}{\partial x_n} \\ \dfrac{\partial f_2}{\partial x_1} & \dfrac{\partial f_2}{\partial x_2} & \cdots & \dfrac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \dfrac{\partial f_n}{\partial x_1} & \dfrac{\partial f_n}{\partial x_2} & \cdots & \dfrac{\partial f_n}{\partial x_n} \end{bmatrix}. \tag{3.30}$$

3.4 STATISTICAL PROPERTIES OF RANDOM PROCESSES (RPs)

3.4.1 Random Processes

An RV was defined as a function *x(s)* defined for each outcome of an experiment identified by s. Now if we assign to each outcome s a time function *x(t, s)*, we obtain a family of functions called *random processes* or *stochastic processes*. An RP is called *discrete* if its argument is a discrete variable set as

$$x(k, s), \quad k = 1, 2 \ldots. \tag{3.31}$$

It is clear that the value of an RP $x(t)$ at any particular time $t = t_0$, namely $x(t_0, s)$, is an RV [or a random vector if $x(t_0, s)$ is vector valued].

3.4.2 Mean, Correlation, and Covariance

Let $x(t)$ be an n-vector RP. Its mean is

$$\mathrm{E}x(t) = \int_{-\infty}^{\infty} x(t)p[x(t)]\,dx(t), \tag{3.32}$$

which can be expressed elementwise as

$$\mathrm{E}x_i(t) = \int_{-\infty}^{\infty} x_i(t)p[x_i(t)]\,dx(t), \quad i = 1 \ldots n.$$

For a random sequence (i.e., an RP in discrete time), the integral is replaced by a sum.

The *correlation* of the vector-valued process $x(t)$ is defined by

$$\mathrm{E}\langle x(t_1)x^{\mathrm{T}}(t_2)\rangle = \begin{bmatrix} \mathrm{E}\langle x(t_1)x_1(t_2)\rangle & \cdots & \mathrm{E}\langle x_1(t_1)x_n(t_2)\rangle \\ \vdots & \ddots & \vdots \\ \mathrm{E}\langle x_n(t_1)x_1(t_2)\rangle & \cdots & \mathrm{E}\langle x_n(t_1)x_n(t_2)\rangle \end{bmatrix}, \tag{3.33}$$

where

$$Ex_i(t_1)x_j(t_2) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x_i(t_1)x_j(t_2)p[x_i(t_1), x_j(t_2)]dx_i(t_1)dx_j(t_2). \tag{3.34}$$

The *covariance* of $x(t)$ is defined by

$$\begin{aligned} &\mathrm{E}\langle[x(t_1) - \mathrm{E}x(t_1)][x(t_2) - \mathrm{E}x(t_2)]^{\mathrm{T}}\rangle \\ &\qquad = \mathrm{E}\langle x(t_1)x^{\mathrm{T}}(t_2)\rangle - \mathrm{E}\langle x(t_1)\rangle\mathrm{E}\langle x^{\mathrm{T}}(t_2)\rangle. \end{aligned} \tag{3.35}$$

When the process $x(t)$ has zero mean (i.e., $\mathrm{E}\,x(t) = 0$ for all t), its correlation and covariance are equal.

The correlation matrix of two RPs $x(t)$, an n-vector, and $y(t)$, an m-vector, is given by an $n \times m$ matrix

$$\mathrm{E}x(t_1)y^{\mathrm{T}}(t_2), \tag{3.36}$$

where

$$\mathrm{E}x_i(t_1)y_j(t_2) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x_i(t_1)y_j(t_2)p[x_i(t_1),y_j(t_2)]\,dx_i(t_1)dy_j(t_2). \tag{3.37}$$

Similarly, the cross-covariance $n \times m$ matrix is

$$\mathrm{E}\langle[x(t_1) - \mathrm{E}x(t_1)][y(t_2) - \mathrm{E}y(t_2)]^{\mathrm{T}}\rangle. \tag{3.38}$$

3.4.3 Orthogonal Processes and White Noise

Two RPs, $x(t)$ and $y(t)$, are called *uncorrelated* if their cross-covariance matrix is identically zero for all t_1 and t_2:

$$\mathrm{E}\langle[x(t_1) - \mathrm{E}\langle x(t_1)\rangle][y(t_2) - \mathrm{E}\langle\, y(t_2)\rangle]^{\mathrm{T}}\rangle = 0. \tag{3.39}$$

The processes $x(t)$ and $y(t)$ are called *orthogonal* if their correlation matrix is identically zero:

$$\mathrm{E}\langle x(t_1)y^{\mathrm{T}}(t_2)\rangle = 0. \tag{3.40}$$

The RP $x(t)$ is called uncorrelated if

$$\mathrm{E}\langle [x(t_1) - \mathrm{E}\langle x(t_1)\rangle][x(t_2) - \mathrm{E}\langle x(t_2)\rangle]^{\mathrm{T}}\rangle = Q(t_1,t_2)\delta(t_1 - t_2), \tag{3.41}$$

where $\delta(t)$ is the Dirac delta "function"[7] (actually, a generalized function), defined by

$$\int_a^b \delta(t)\,dt = \begin{cases} 1 & \text{if } a \le 0 \le b, \\ 0 & \text{otherwise.} \end{cases} \tag{3.42}$$

Similarly, a random sequence x_k is called uncorrelated if

$$\mathrm{E}\langle [x_k - \mathrm{E}\langle x_k\rangle][x_j - \mathrm{E}\langle x_j\rangle]^{\mathrm{T}}\rangle = Q(k,j)\Delta(k - j), \tag{3.43}$$

where $\Delta(\cdot)$ is the Kronecker delta function,[8] defined by

$$\Delta(k) = \begin{cases} 1 & \text{if } k = 0 \\ 0 & \text{otherwise.} \end{cases} \tag{3.44}$$

A white-noise process or sequence is an example of an uncorrelated process or sequence.

A process $x(t)$ is considered independent if for any choice of distinct times t_1, t_2, $\ldots$, t_n, the RVs $x(t_1)$, $x(t_2)$, $\ldots$, $x(\mathrm{t}_n)$ are independent. That is,

$$p_{x(t_1)}, \ldots, p_{x(t_n)}(s_1, \ldots, s_n) = \prod_{i=1}^{n} p_{x(t_i)}(s_i). \tag{3.45}$$

Independence (all of the moments) implies no correlation (which restricts attention to the second moments), but the opposite implication is not true, except in such special cases as Gaussian processes (see Section 3.2.3). Note that *whiteness* means *uncorrelated* in time rather than *independent* in time (i.e., including all moments), although this distinction disappears for the important case of white Gaussian processes (see Chapter 4).

[7]Named for the English physicist Paul Adrien Maurice Dirac (1902–1984).
[8]Named for the German mathematician Leopold Kronecker (1823–1891).

3.4.4 Strict-Sense and Wide-Sense Stationarity

The RP $x(t)$ (or random sequence x_k) is called *strict-sense stationary* if all its statistics (meaning $p[x(t_1), x(t_2), \ldots]$) are invariant with respect to shifts of the time origin:

$$p(x_1, x_2, \ldots, x_n, t_1, \ldots, t_n) = p(x_1, x_2, \ldots, x_n, t_1 + \varepsilon, t_2 + \varepsilon, \ldots, t_n + \varepsilon). \tag{3.46}$$

The RP $x(t)$ (or x_k) is called *wide-sense stationary* (WSS) (or *weak-sense stationary*) if

$$\mathrm{E}\langle x(t)\rangle = c \text{ (a constant)} \tag{3.47}$$

and

$$\mathrm{E}\langle x(t_1)x^{\mathrm{T}}(t_2)\rangle = Q(t_2 - t_1) = Q(\tau), \tag{3.48}$$

where Q is a matrix with each element depending only on the difference $t_2 - t_1 = \tau$. Therefore, when $x(t)$ is stationary in the weak sense, it implies that its first- and second-order statistics are independent of time origin, while strict stationarity by definition implies that statistics of all orders are independent of the time origin.

3.4.5 Ergodic Random Processes

A process is considered ergodic[9] if all of its statistical parameters, mean, variance, and so on, can be determined from arbitrarily chosen member functions. A sampled function $x(t)$ is ergodic if its time-averaged statistics equal the ensemble averages.

3.4.6 Markov Processes and Sequences

An RP $x(t)$ is called a *Markov process*[10] if its future state distribution, conditioned on knowledge of its present state, is not improved by knowledge of previous states:

$$p\{x(t_i)|x(\tau);\tau < t_{i-1}\} = p\{x(t_i)|x(t_{i-1})\}, \tag{3.49}$$

where the times $t_1 < t_2 < t_3 < \cdots < t_i$.

[9]The term *ergodic* came originally from the development of statistical mechanics for thermodynamic systems. It is taken from the Greek words for *energy* and *path*. The term was applied by the American physicist Josiah Willard Gibbs (1839–1903) to the time history (or path) of the state of a thermodynamic system of constant energy. Gibbs had assumed that a thermodynamic system would eventually take on all possible states consistent with its energy. It was shown to be impossible from function-theoretic considerations in the nineteenth century. The so-called ergodic hypothesis of James Clerk Maxwell (1831–1879) is that the temporal means of a stochastic system are equivalent to the ensemble means. The concept was given firmer mathematical foundations by George David Birkhoff and John von Neumann around 1930 and by Norbert Wiener in the 1940s.

[10]Named after the Russian mathematician Andrei Andreyevich Markov (1856–1922), who first developed many of the concepts and the related theory.

Similarly, a random sequence (RS) x_k is called a *Markov sequence* if

$$p(x_i|x_k;\, k \le i-1) = p\{x_i|x_{i-1}\}. \tag{3.50}$$

The solution to a general first-order differential or difference equation with an independent process (uncorrelated normal RP) as a forcing function is a Markov process. That is, if $x(t)$ and x_k are n-vectors satisfying

$$\dot{x}(t) = F(t)\ x(t) + G(t)\ w(t) \tag{3.51}$$

or

$$x_k = \Phi_{k-1}\ x_{k-1} + G_{k-1}\ w_{k-1}, \tag{3.52}$$

where $w(t)$ and w_{k-1} are r-dimensional independent RPs and sequences, the solutions $x(t)$ and x_k are then vector Markov processes and sequences, respectively.

3.4.7 Gaussian RPs

An n-dimensional RP $x(t)$ is called Gaussian (or normal) if its probability density function is Gaussian, as given by the formulas of Section 3.2.3, with covariance matrix

$$P = \mathrm{E}\langle|x(t) - \mathrm{E}\langle x(t)\rangle[x(t) - \mathrm{E}\langle x(t)\rangle]^{\mathrm{T}}\rangle \tag{3.53}$$

for the RV x.

Gaussian RPs have some useful properties:

1. A Gaussian RP $x(t)$ is WSS—and stationary in the strict sense.
2. Orthogonal Gaussian RPs are independent.
3. Any linear function of jointly Gaussian RPs results in another Gaussian RP.
4. All statistics of a Gaussian RP are completely determined by its first- and second-order statistics.

3.4.8 Simulating Multivariate Gaussian Processes

Cholesky decomposition methods are discussed in Chapter 6 and Appendix B. Here, we show how these methods can be used to generate uncorrelated pseudorandom vector sequences with zero mean (or any specified mean) and a specified covariance P.

There are many programs that will generate pseudorandom sequences of uncorrelated Gaussian scalars $\{s_i \mid i = 1, 2, 3, \ldots\}$ with zero mean and unit variance:

$$\mathrm{E}\langle s_i\rangle \in \mathcal{N}(0,1) \text{ for all } i, \tag{3.54}$$

$$\mathrm{E}\langle s_i s_j\rangle = \begin{cases} 0 & \text{if} \quad i \ne j, \\ 1 & \text{if} \quad i = j \end{cases} \tag{3.55}$$

These can be used to generate sequences of Gaussian n-vectors u_k with mean zero and covariance I_m:

$$u_k = [s_{nk+1} s_{nk+2} s_{nk+3} \ldots s_{n(k+1)}]^{\mathrm{T}}, \tag{3.56}$$

$$\mathrm{E}\langle u_k \rangle = 0, \tag{3.57}$$

$$\mathrm{E}\langle u_k u_k^{\mathrm{T}} \rangle = I_n. \tag{3.58}$$

These vectors, in turn, can be used to generate a sequence of n-vectors w_k with zero mean and covariance P. For that purpose, let

$$CC^{\mathrm{T}} = P \tag{3.59}$$

be a Cholesky decomposition of P, and let the sequence of n-vectors w_k be generated according to the rule

$$w_k = Cu_k. \tag{3.60}$$

Then the sequence of vectors $\{w_0, w_1, w_2, \ldots\}$ will have mean

$$\mathrm{E}\langle w_k \rangle = C\mathrm{E}\langle u_k \rangle \tag{3.61}$$

$$= 0 \tag{3.62}$$

(an n-vector of zeros) and covariance

$$\mathrm{E}\langle w_k w_k^{\mathrm{T}} \rangle = \mathrm{E}\langle Cu_k (Cu_k)^{\mathrm{T}} \rangle \tag{3.63}$$

$$= CI_n C^{\mathrm{T}} \tag{3.64}$$

$$= P. \tag{3.65}$$

The same technique can be used to obtain pseudorandom Gaussian vectors with a given mean v by adding v to each w_k. These techniques are used in simulation and Monte Carlo analysis of stochastic systems.

3.4.9 Power Spectral Density (PSD)

Let $x(t)$ be a zero-mean scalar stationary RP with autocovariance $\psi_x(\tau)$,

$$\mathrm{E}\langle x(t)x(t+\tau) \rangle = \psi_x(\tau). \tag{3.66}$$

The PSD is defined as

$$\Psi_x(\omega) = \int_{-\infty}^{\infty} \psi_x(\tau) e^{-j\omega\tau}\, d\tau \tag{3.67}$$

and the inverse transform as

$$\psi_x(\tau) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \Psi_x(\omega) e^{j\omega\tau}\, d\omega. \tag{3.68}$$

The following are properties of autocovariance functions:

1. Autocovariance functions are symmetrical ("even" functions).
2. An autocovariance function attains its maximum value at the origin.
3. Its Fourier transform is nonnegative (greater than or equal to zero).

These properties are satisfied by valid autocovariance functions.

Setting $\tau = 0$ in Equation (3.68) gives

$$\mathrm{E}_t\langle x^2(t)\rangle = \psi_x(0) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \Psi_x(\omega)\, d\omega. \tag{3.69}$$

Because of property 1 of the autocovariance function,

$$\Psi_x(\omega) = \Psi_x(-\omega); \tag{3.70}$$

that is, the PSD is a symmetric function of frequency.

Example 3.3 (PSD for an Exponentially Correlated Process) The exponentially correlated process diagrammed in Figure 3.5(a) has an autocovariance function of the general form

$$\psi_x(t) = \sigma^2 e^{-|t|/\tau},$$

where σ^2 is the mean-squared process amplitude and τ is the autocorrelation time constant. Its PSD is the Fourier transform of its autocovariance function,

$$\begin{aligned}\Psi_x(\omega) &= \int_{-\infty}^{0} \sigma^2 e^{t/\tau} e^{-j\omega t}\, dt + \int_{0}^{\infty} \sigma^2 e^{-t/\tau} e^{-j\omega t}\, dt \\ &= \sigma^2 \left(\frac{1}{1/\tau - j\omega} + \frac{1}{1/\tau + j\omega} \right) = \frac{2\sigma^2 \tau}{1/\tau^2 + \omega^2},\end{aligned}$$

the shape of which is illustrated in Figure 3.5(b) as a log-log plot. White noise passed through a low-pass RC filter with resistance R (in ohms) and capacitance C (in farads) would have a filter output spectrum of this shape, with the "knee" at $\omega = 1/\tau = 1/(R \times C)$.

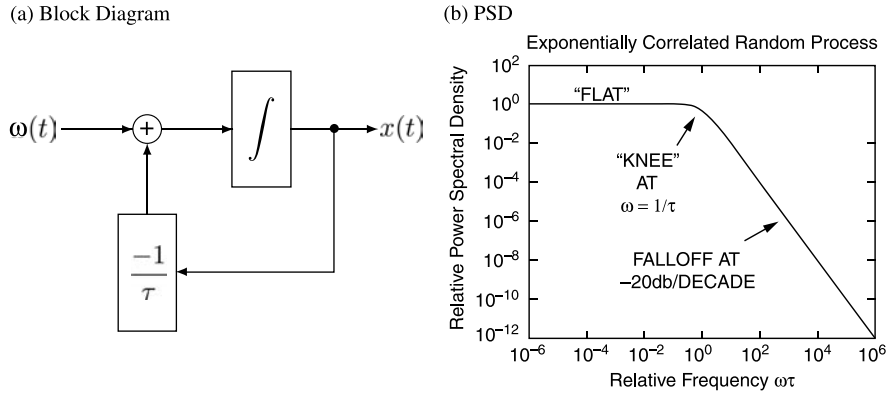


Figure 3.5 Exponentially correlated RP.

Example 3.4 [Underdamped Harmonic Oscillator as a Shaping Filter] This is an example of a second-order Markov process generated by passing WSS white noise with zero mean and unit variance through a second-order "shaping filter" with the dynamic model of a harmonic resonator. The dynamic model for a damped harmonic oscillator was introduced in Examples 2.2–2.3 and 2.6–2.7 of Chapter 2 and will be used again in Chapters 4 and 7.

The transfer function of the dynamic system is

$$H(s) = \frac{as + b}{s^2 + 2\zeta\omega_n s + \omega_n^2}.$$

Definitions of ζ, ω_n, and s are the same as in Example 2.7. The state-space model of $H(s)$ is given as

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\omega_n^2 & -2\zeta\omega_n \end{bmatrix} \begin{bmatrix} x_{1(t)} \\ x_2(t) \end{bmatrix} + \begin{bmatrix} a \\ b - 2a\zeta\omega_n \end{bmatrix} w(t),$$
$$z(t) = x_1(t) = x(t).$$

The general form of the autocorrelation is

$$\psi_x(\tau) = \frac{\sigma^2}{\cos\theta} e^{-\zeta\omega_n|\tau|} \cos\left(\sqrt{1-\zeta^2}\ \omega_n|\tau| - \theta\right).$$

In practice, σ^2, θ, ζ, and ω_n are chosen to fit empirical data (see Problem 3.13). The PSD corresponding to the $\psi_x(\tau)$ will have the form

$$\Psi_x(\omega) = \frac{a^2\ \omega^2 + b^2}{\omega^4 + 2\ \omega_n^2(2\zeta^2 - 1)\ \omega^2 + \omega_n^4}.$$

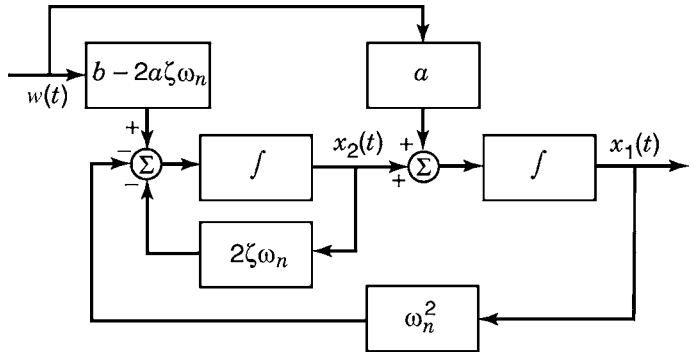


Figure 3.6 Diagram of a second-order Markov process.

(The peak of this PSD will not be at the "natural" (undamped) frequency ω_n, but at the "resonant" frequency defined in Example 2.6.)

The block diagram corresponding to the state-space model is shown in Figure 3.6.

The *mean power* of a scalar RP is given by the equations

$$\mathrm{E}_t\langle x^2(t)\rangle = \lim_{T\to\infty}\int_{-T}^{\mathrm{T}} x^2(t)\,dt \tag{3.71}$$

$$= \frac{1}{2\pi}\int_{-\infty}^{\infty} \Psi_x(\omega)\,d\omega \tag{3.72}$$

$$= \sigma^2. \tag{3.73}$$

The cross PSD between an RP x and an RP y is given by the formula

$$\Psi_{xy}(\omega) = \int_{-\infty}^{\infty} \psi_{xy}(\tau)e^{-j\omega\tau}\,d\tau. \tag{3.74}$$

3.5 LINEAR RP MODELS

Linear system models of the type illustrated in Figure 3.7 are defined by the equation

$$y(t) = \int_{-\infty}^{\infty} x(\tau)h(t,\tau)\,d\tau, \tag{3.75}$$

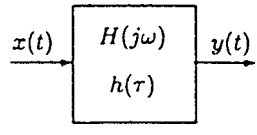


Figure 3.7 Block diagram representation of a linear system.

where $x(t)$ is input and $h(t, \tau)$ is the linear system *weighting function* (see Figure 3.7). If the system is time invariant (i.e., h does not depend on t), then Equation (3.75) can be written as

$$y(t) = \int_0^\infty h(\tau)x(t - \tau)\, d\tau. \tag{3.76}$$

This type of integral is called a *convolution integral*, and $h(\tau)$ is called its *kernel function*. Manipulation of Equation (3.76) leads to relationships between autocovariance functions of $x(t)$ and $y(t)$,

$$\psi_y(\tau) = \int_0^\infty d\tau_1 h(\tau_1) \int_0^\infty d\tau_2 h(\tau_2)\psi_x(\tau + \tau_1 - \tau_2) \text{ (autocovariance)}, \tag{3.77}$$

$$\psi_{xy}(\tau) = \int_0^\infty h(\tau_1)\psi_x(\tau - \tau_1)\, d\tau_1 \text{ (cross-covariance)}, \tag{3.78}$$

and spectral relationships

$$\Psi_{xy}(\omega) = H(j\omega)\Psi_x(\omega) \text{ (cross-spectrum)}, \tag{3.79}$$

$$\Psi_y(\omega) = |H(j\omega)|^2\Psi_x(\omega) \text{ (PSD)}, \tag{3.80}$$

where H is the *system transfer function* (also shown in Figure 3.7), defined by the Laplace transform of $h(\tau)$ as

$$\int_0^\infty h(\tau)e^{s\tau}\, d\tau = H(s) = H(j\omega), \tag{3.81}$$

where $s = j\omega$ and $j = \sqrt{-1}$.

3.5.1 Stochastic Differential Equations for RPs

The Calculus of Stochastic Differential Equations Differential equations involving RPs are called *stochastic differential equations.* Introducing RPs as inhomogeneous terms in ordinary differential equations has ramifications beyond the level of rigor that will be followed here, and the reader should be aware of them. The problem is that RPs are not integrable functions in the conventional (Riemann) calculus. The resolution of this problem requires foundational modifications of the calculus to obtain many of the results presented. The Riemann integral of the "ordinary" calculus must be modified to what is called the *Itô calculus*. The interested reader will find these issues treated more rigorously in the books by Bucy and Joseph [56], Itô [114], and Jazwinski [116].

Linear Stochastic Differential Equations A linear stochastic differential equation as a model of an RP with initial conditions has the general form

$$\begin{aligned} \dot{x}(t) &= F(t)x(t) + G(t)w(t) + C(t)u(t), \\ z(t) &= H(t)x(t) + v(t) + D(t)u(t), \end{aligned} \tag{3.82}$$

where the variables are defined as

$x(t) = n \times 1$ state vector,
$z(t) = \ell \times 1$ measurement vector,
$u(t) = r \times 1$ deterministic input vector,
$F(t) = n \times n$ time-varying dynamic coefficient matrix,
$C(t) = n \times r$ time-varying input coupling matrix,
$H(t) = \ell \times n$ time-varying measurement sensitivity matrix,
$D(t) = \ell \times r$ time-varying output coupling matrix,
$G(t) = n \times r$ time-varying process noise coupling matrix,
$w(t) = r \times 1$ zero-mean uncorrelated "plant noise" process,
$v(t) = \ell \times 1$ zero-mean uncorrelated "measurement noise" process

and the expected values as

$$\begin{aligned} \mathrm{E}\langle w(t) \rangle &= 0, \\ \mathrm{E}\langle v(t) \rangle &= 0, \\ \mathrm{E}\langle w(t_1)\, w^{\mathrm{T}}(t_2) \rangle &= Q(t_1)\delta(t_2 - t_1), \\ \mathrm{E}\langle v(t_1)\, v^{\mathrm{T}}(t_2) \rangle &= R(t_1)\delta(t_2 - t_1). \\ \mathrm{E}\langle w(t_1)\, v^{\mathrm{T}}(t_2) \rangle &= M(t_1)\delta(t_2 - t_1) \end{aligned}$$

$$\delta(t) = \begin{cases} 1, & t = 0, \\ 0, & t \neq 0. \end{cases} \quad \text{(Dirac delta function)}$$

The symbols Q, R, and M represent $r \times r$, $\ell \times \ell$, and $r \times \ell$ matrices, respectively, and δ represents the Dirac delta "function" (a measure). The values over time of the variable $x(t)$ in the differential equation model define vector-valued Markov processes. This model is a fairly accurate and useful representation for many real-world processes, including stationary Gaussian and nonstationary Gaussian processes, depending on the statistical properties of the RVs and the temporal properties of the deterministic variables. [The function $u(t)$ usually represents a known control input. For the rest of the discussion in this chapter, we will assume that $u(t) = 0$.]

Example 3.5 (PSD of Exponentially Correlated RP) Continuing with Example 3.3, let the RP $x(t)$ be a zero-mean stationary normal RP having autocovariance

$$\psi_x(\tau) = \sigma^2 e^{-\alpha|\tau|}. \tag{3.83}$$

The corresponding PSD is

$$\Psi_x(\omega) = \frac{2\sigma^2\alpha}{\omega^2 + \alpha^2}. \tag{3.84}$$

This type of RP can be modeled as the output of a linear system with input $w(t)$, a zero-mean white Gaussian noise with PSD equal to unity. Using Equation 3.80, one can derive the transfer function $H(j\omega)$ for the following model:

$$H(j\omega)H(-j\omega) = \frac{\sqrt{2\alpha}\sigma}{\alpha + j\omega} \cdot \frac{\sqrt{2\alpha}\sigma}{\alpha - j\omega}.$$

Take the stable portion of this system transfer function as

$$H(s) = \frac{\sqrt{2\alpha}\sigma}{s + \alpha}, \tag{3.85}$$

which can be represented as

$$\frac{x(s)}{w(s)} = \frac{\sqrt{2\alpha}\sigma}{s + \alpha}, \tag{3.86}$$

By taking the inverse Laplace transform of both sides of this last equation, one can obtain the following sequence of equations:

$$\begin{aligned} \dot{x}(t) + \alpha x(t) &= \sqrt{2\alpha}\sigma w(t), \\ \dot{x}(t) &= -\alpha x(t) + \sqrt{2\alpha}\sigma w(t), \\ z(t) &= x(t), \end{aligned}$$

with $\sigma_x^2(0) = \sigma^2$. The parameter $1/\alpha$ is the *correlation time* of the process.

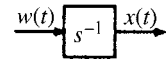
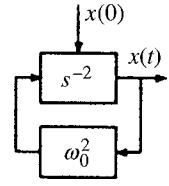
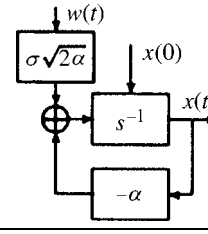
The block diagram representation of the process in Example 3.5 is shown in Table 3.1. This is called a *shaping filter*. Some other examples of differential equation models are also given in Table 3.1.

3.5.2 Discrete Model of a Random Sequence (RS)

RPs in discrete time are also called *random sequences.* A vector-valued discrete-time recursive equation for modeling an RS with initial conditions can be given in the form

$$\begin{aligned} x_k &= \Phi_{k-1}x_{k-1} + G_{k-1}\, w_{k-1} + \Gamma_{k-1}u_{k-1}, \\ z_k &= H_k x_k + \upsilon_k + D_k u_k. \end{aligned} \tag{3.87}$$

TABLE 3.1 System Models of RPs

RP	Autocovariance Function and PSD	Shaping Filter Diagram	State-Space Model
White noise	$\psi_x(\tau) = \sigma^2\ \delta^2(\tau)$ $\Psi_x(\omega) = \sigma^2$	None	Measurement noise $v(t)$
Random walk	$\psi_x(\tau) =$ (undefined) $\Psi_x(\omega) \propto \sigma^2/\omega^2$		$\dot{x} = w(t)$ $\sigma_x^2(0) = 0$
Random constant	$\psi_x(\tau) = \sigma^2$ $\Psi_x(\omega) = 2\pi\sigma^2\delta(\omega)$	None	$\dot{x} = 0$ $\sigma_x^2(0) = \sigma^2$
Sinusoid	$\psi_x(\tau) = \sigma^2\cos(\omega_0\ \tau)$ $\Psi_x(\omega) = \pi\sigma^2\delta(\omega - \omega_0) + \pi\sigma^2\delta(\omega + \omega_0)$		$\dot{x} = \begin{bmatrix} 0 & 1 \\ -\omega_0^2 & 0 \end{bmatrix} x$ $P(0) = \begin{bmatrix} \sigma^2 & 0 \\ 0 & 0 \end{bmatrix}$
Exponentially correlated	$\psi_x(\tau) = \sigma^2 e^{-\alpha\lvert\tau\rvert}$ $\Psi_x(\omega) = \dfrac{2\sigma^2\alpha}{\omega^2 + \alpha^2}$ $\frac{1}{\alpha}$ = correlation time		$\dot{x} = -\alpha x + \sigma\sqrt{2\alpha}w(t)$ $\sigma_x^2(0) = \sigma^2$

This is the complete model with deterministic inputs u_k as discussed in Chapter 2, Equations 2.31 and 2.32, and RS noise w_k and v_k, as described in Chapter 4 equations:

$$
\begin{aligned}
x_k &= n \times 1 \text{ state vector} \\
z_k &= \ell \times 1 \text{ measurement vector} \\
u_k &= r \times 1 \text{ deterministic input vector} \\
\Phi_{k-1} &= n \times n \text{ time-varying matrix} \\
G_{k-1} &= n \times r \text{ time-varying matrix} \\
H_k &= \ell \times n \text{ time-varying matrix} \\
D_k &= \ell \times r \text{ time-varying matrix} \\
\Gamma_{k-1} &= n \times r \text{ time-varying matrix} \\
\mathrm{E}\langle w_k \rangle &= 0 \\
\mathrm{E}\langle v_k \rangle &= 0 \\
\mathrm{E}\langle w_{k_1} w_{k_2}^{\mathrm{T}} \rangle &= Q_{k_1}\Delta(k_2 - k_1) \qquad \text{(Kronecker delta)} \\
\mathrm{E}\langle v_{k_1} v_{k_2}^{\mathrm{T}} \rangle &= R_{k_1}\Delta(k_2 - k_1) \\
\mathrm{E}\langle w_{k_1} v_{k_2}^{\mathrm{T}} \rangle &= M_{k_1}\Delta(k_2 - k_1)
\end{aligned}
$$

Example 3.6 (Expontially Correlated RS) Let the $\{x_k\}$ be a zero-mean stationary Gaussian RS with autocorrelation

$$\psi_x(k_2 - k_1) = \sigma^2 e^{-\alpha|k_2 - k_1|}.$$

This type of RS can be modeled as the output of a linear system, with input w_k being zero-mean white Gaussian noise with PSD equal to unity.

A difference equation model for this type of process can be defined as

$$x_k = \Phi x_{k-1} + G w_{k-1}, \quad z_k = x_k. \tag{3.88}$$

In order to use this model, we need to solve for the unknown parameters Φ and G as functions of the parameter α. To do so, we first multiply Equation (3.88) by x_{k-1} on both sides and take the expected values to obtain the equations

$$\begin{aligned} \mathrm{E}\langle x_k x_{k-1}\rangle &= \Phi \mathrm{E}\langle x_{k-1} x_{k-1}\rangle + G\mathrm{E}\langle w_{k-1} x_{k-1}\rangle, \\ \sigma^2 e^{-\alpha} &= \Phi \sigma^2, \end{aligned}$$

assuming the w_k are uncorrelated and $\mathrm{E}\langle w_K\rangle = 0$, so that $\mathrm{E}\langle w_{k-1}\ x_{k-1}\rangle = 0$. One obtains the solution

$$\Phi = e^{-\alpha}. \tag{3.89}$$

Next, square the state variable defined by Equation (3.88) and take its expected value:

$$\mathrm{E}\langle x_k^2\rangle = \Phi^2 \mathrm{E}\langle x_{k-1} x_{k-1}\rangle + G^2 \mathrm{E}\langle w_{k-1} w_{k-1}\rangle, \tag{3.90}$$

$$\sigma^2 = \sigma^2 \Phi^2 + G^2, \tag{3.91}$$

because the variance $\mathrm{E}\langle w_{k-1}^2\rangle = 1$ and the parameter $G = \sigma\sqrt{1 - e^{-2\alpha}}$.

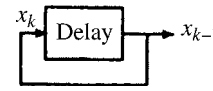
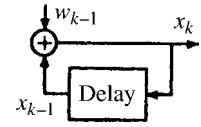
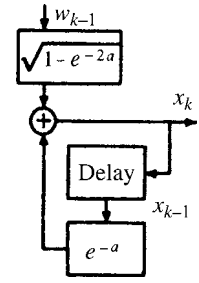
The complete model is then

$$x_k = e^{-\alpha} x_{k-1} + \sigma\sqrt{1 - e^{-2\alpha}}\, w_{k-1}$$

with $\mathrm{E}\langle w_k\rangle = 0$ and $\mathrm{E}\langle w_{k_1} w_{k_2}\rangle = \Delta(k_2 - k_1)$.

The dynamic process model derived in Example 3.6 is called a *shaping filter*. Block diagrams of this and other shaping filters are given in Table 3.2, along with their difference equation models.

TABLE 3.2 Stochastic System Models of Discrete RSs

Process Type	Autocovariance	Block Diagram	State-Space Model
Random constant	$\psi_x(k_1-k_2)=\sigma^2$		$x_k = x_{k-1}$ $\sigma_x^2(0)=\sigma^2$
Random walk	$\to +\infty$		$x_k = x_{k-1} + w_{k-1}$ $\sigma_x^2(0)=0$
Exponentially correlated noise	$\psi_x(k_2-k_1)=\sigma^2$ $e^{-\alpha\lvert k_2-k_1\rvert}$		$x_k = e^{-\alpha} x_{k-1} + \sigma\sqrt{1-e^{-2\alpha}}\, w_{k-1}$ $\sigma_x^2(0)=\sigma^2$

3.5.3 Autoregressive Processes and Linear Predictive Models

A *linear predictive model* for a signal is a representation in the form

$$\acute{x}_{k+1} = \sum_{i=1}^{n} a_i \acute{x}_{k-i+1} + \acute{u}_k, \tag{3.92}$$

where $\acute{u}_k$ is the *prediction error*. Successive samples of the signal are predicted as linear combinations of the n previous values.

An *autoregressive process* has the same formulation, except that $\acute{u}_k$ is a white Gaussian noise process. Note that this formula for an autoregressive process can be rewritten in state transition matrix form as

$$\begin{bmatrix} \acute{x}_{k+1} \\ \acute{x}_k \\ \acute{x}_{k_1} \\ \vdots \\ \acute{x}_{k-n+2} \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & \cdots & a_{n-1} & a_n \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \begin{bmatrix} \acute{x}_k \\ \acute{x}_{k-1} \\ \acute{x}_{k-2} \\ \vdots \\ \acute{x}_{k-n+1} \end{bmatrix} + \begin{bmatrix} \acute{u}_k \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \tag{3.93}$$

$$x_{k+1} = \Phi x_k + w_k, \tag{3.94}$$

where the "state" is the n-vector of the last n samples of the signal and the covariance matrix Q_k of the associated process noise w_k will be filled with zeros, except for the term $q_{11} = \mathrm{E}\,\langle u_k^2 \rangle$.

3.6 SHAPING FILTERS AND STATE AUGMENTATION

Shaping Filters The focus of this section is nonwhite noise models for stationary RPs. For many physical systems encountered in practice, it may not be justified to assume that all noises are white Gaussian noise processes. It can be useful to generate an autocorrelation function or PSD from real data and then develop an appropriate noise model using differential or difference equations. These models are called *shaping filters*, a concept introduced in works by Hendrik Wade Bode (1905–1982) and Claude Elwood Shannon (1916–2001) [41] and by Lotfi Asker Zadeh and John Ralph Ragazzini (1912–1988) [293]. They are filters driven by noise with a flat spectrum (white noise processes), which they shape to represent the spectrum of the actual system. It was shown in the previous section that a linear time-invariant system (shaping filter) driven by WSS white Gaussian noise provides such a model. The state vector can be "augmented" by appending to it the state vector components of the shaping filter, with the resulting model having the form of a linear dynamic system driven by white noise.

3.6.1 Correlated Process Noise Models

Shaping Filters for Dynamic Disturbance Noise Let a system model be given by

$$\dot{x}(t) = F(t)x(t) + G(t)w_1(t), \quad z(t) = H(t)x(t) + v(t), \tag{3.95}$$

where $w_1(t)$ is nonwhite, for example, correlated Gaussian noise. As given in the previous section, $v(t)$ is a zero-mean white Gaussian noise. Suppose that $w_1(t)$ can be modeled by a linear shaping filter[11]:

$$\begin{aligned} \dot{x}_{\mathrm{SF}}(t) &= F_{\mathrm{SF}}(t)x_{\mathrm{SF}}(t) + G_{\mathrm{SF}}(t)w_2(t) \\ w_1(t) &= H_{\mathrm{SF}}(t)x_{\mathrm{SF}}(t) \end{aligned} \tag{3.96}$$

where SF denotes the shaping filter and $w_2(t)$ is zero mean white Gaussian noise. Now define a new augmented state vector

$$X(t) = [x^{\mathrm{T}}(t), x_{\mathrm{SF}}^{\mathrm{T}}(t)]^{\mathrm{T}}. \tag{3.97}$$

[11] See Example 3.4 for WSS processes.

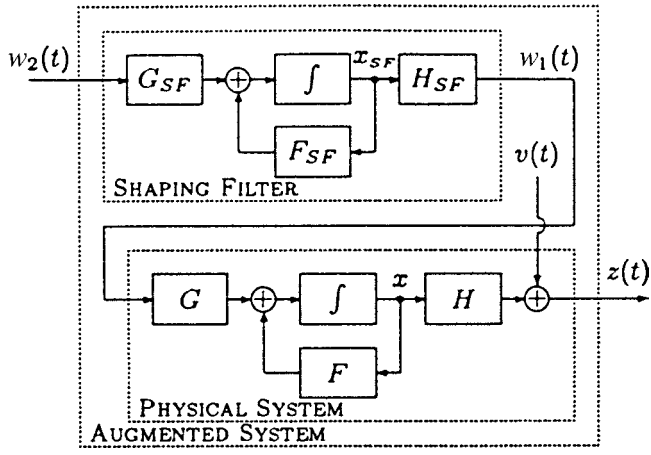


Figure 3.8 Shaping filter model for non-white noise.

Equations (3.95) and (3.96) can be combined into the matrix form

$$\begin{bmatrix} \dot{x}(t) \\ \dot{x}_{\mathrm{SF}}(t) \end{bmatrix} = \begin{bmatrix} F(t) & G(t)H_{\mathrm{SF}}(t) \\ 0 & F_{\mathrm{SF}}(t) \end{bmatrix} \begin{bmatrix} x(t) \\ x_{\mathrm{SF}}(t) \end{bmatrix} + \begin{bmatrix} 0 \\ G_{\mathrm{SF}}(t) \end{bmatrix} w_2(t), \tag{3.98}$$

$$\dot{X}(t) = F_T(t)X(t) + G_T(t)w_2(t), \tag{3.99}$$

and the output equation can be expressed in a compatible format as

$$z(t) = [H(t)\ 0] \begin{bmatrix} x(t) \\ x_{\mathrm{SF}}(t) \end{bmatrix} + v(t) \tag{3.100}$$

$$= H_T(t)X(t) + v(t). \tag{3.101}$$

This augmented system given by Equations 3.99 and 3.101 is a linear differential equation model driven by white Gaussian noise. (See Figure 3.8 for a non-white-noise model.)

3.6.2 Correlated Measurement Noise Models

Shaping Filters for Measurement Noise A similar development is feasible for the case of time-correlated measurement noise $v_1(t)$:

$$\begin{aligned} \dot{x}(t) &= F(t)x(t) + G(t)w(t), \\ z(t) &= H(t)x(t) + v_1(t). \end{aligned} \tag{3.102}$$

In this case, let $v_2(t)$ be zero-mean white Gaussian noise and let the measurement noise $v_1(t)$ be modeled by

$$\begin{aligned} \dot{x}_{\mathrm{SF}}(t) &= F_{\mathrm{SF}}(t)x_{\mathrm{SF}}(t) + G_{\mathrm{SF}}(t)v_2(t), \\ v_1(t) &= H_{\mathrm{SF}}(t)x_{\mathrm{SF}}(t). \end{aligned} \tag{3.103}$$

The total augmented system is given by

$$\begin{aligned} \begin{bmatrix} \dot{x}(t) \\ \dot{x}_{\mathrm{SF}}(t) \end{bmatrix} &= \begin{bmatrix} F(t) & 0 \\ 0 & F_{\mathrm{SF}}(t) \end{bmatrix} \begin{bmatrix} x(t) \\ x_{\mathrm{SF}}(t) \end{bmatrix} + \begin{bmatrix} G(t) & 0 \\ 0 & G_{\mathrm{SF}}(t) \end{bmatrix} \begin{bmatrix} w(t) \\ v_2(t) \end{bmatrix}, \\ z(t) &= [H(t)\ H_{\mathrm{SF}}(t)] \begin{bmatrix} x(t) \\ x_{\mathrm{SF}}(t) \end{bmatrix}. \end{aligned} \tag{3.104}$$

This is in the form of a linear system model driven by white Gaussian noise and an output equation with no input noise.

These systems can be specialized to the WSS process for continuous and discrete cases as by the shaping filters shown in Tables 3.1 and 3.2.

Example 3.7 (Accelerometer Model) The "midpoint acceleration" error for an acceleration sensor (accelerometer) is defined as the effective acceleration error at the midpoint of the sampling period. The associated error model for an accelerometer in terms of unknown parameters of the sensor is as follows:

$$\Delta_{\beta m} = \beta_m \otimes \zeta + b_A + h_A\beta_m + \beta_m^2(FI1 - FX1) + \delta\beta,$$

where

$\Delta_{\beta m}$ = the midpoint acceleration error
$\otimes$ = the cross product (for 3×1 vectors)
ζ = a 3×1 vector representing attitude alignment errors between "platform" axes and computational axes
b_A = a 3×1 vector of unknown accelerometer biases, normalized to the magnitude of gravity

$$h_A = \begin{bmatrix} S_1 & \delta_{12} & \delta_{13} \\ 0 & S_2 & \delta_{23} \\ 0 & 0 & S_3 \end{bmatrix}$$

and

S_i = unknown accelerometer scale factor errors ($i = 1, 2, 3$)
δ_{ij} = unknown accelerometer axes nonorthogonalities

$\delta\beta =$ other error terms, some of which are observable; for practicality in our example, they are not estimated, only compensated for with factory-calibrated values

$FI1 =$ a 3×1 unknown acceleration-squared nonlinearity for acceleration along the accelerometer input axis

$FX1 =$ a 3×1 unknown acceleration-squared nonlinearity for acceleration normal to the accelerometer input axis

$\beta_m =$ a 3×1 vector $[\beta_1, \beta_2, \beta_3]^{\mathrm{T}}$ of midpoint components of acceleration in platform coordinates

$$\beta_m^2 = \begin{bmatrix} \beta_m^2 & 0 & 0 \\ 0 & \beta_2^2 & 0 \\ 0 & 0 & \beta_3^2 \end{bmatrix}$$

The 12×1 accelerometer state vector x^{A} is composed of the subvectors and scalars

$$x^A = \left[\underbrace{b_A^{\mathrm{T}}}_{1\times 3} \; S_1 \; \delta_{12} \; S_2 \; \delta_{13} \; \delta_{23} \; S_3 \; \underbrace{(FX1 - FI1)^{\mathrm{T}}}_{1\times 3} \right]^{\mathrm{T}}.$$

The 12 unknown parameters will be modeled as random walks (see Table 3.1) for the parameter identification problem to be discussed in Chapter 7 (Example 7.6).

Example 3.8 (Gyroscope Calibration Model) A gyroscope drift error model is given as follows:

$$\mathcal{E} = b_g + h_g\omega + U_g\beta + K_g\beta^1 + \big[|\omega|\big]T_g + b_{gt}t + U_{gt}t\beta,$$

where

$b_g =$ a 3×1 vector of unknown gyroscope fixed drift parameters

$h_g =$ a 3×1 matrix containing unknown scale factor (S_{gi}) and linear axes alignment errors (Δ_{ij}) as components ($i, j = 1, 2, 3$)

$$\begin{bmatrix} S_{g1} & \Delta_{12} & \Delta_{13} \\ \Delta_{21} & S_{g2} & \Delta_{23} \\ \Delta_{31} & \Delta_{32} & S_{g3} \end{bmatrix}$$

$T_g =$ a 3×1 vector of unknown nonlinear gyroscope torquer scale factor errors, with elements δS_{gi}

$\big[|\omega|\big] =$ a 3×3 diagonal matrix composed of absolute values of the components of ω (platform inertial angular rate) on the corresponding diagonal element

$U_g =$ a 3×3 matrix of unknown gyroscope mass unbalance parameters ($d_{k,j}$)

$$\begin{bmatrix} d_{I1} & d_{01} & d_{S1} \\ d_{S2} & d_{I2} & d_{02} \\ d_{03} & d_{S3} & d_{I3} \end{bmatrix}$$

indices I, 0, and S denoting input, output, and spin axes, respectively, for each gyroscopes 1, 2, and 3

$K_g =$ a 3×6 matrix of unknown gyroscope compliance (g-squared) errors k_{kji}

$$\begin{bmatrix} k_{II1} & k_{001} & k_{SS1} & I_{I01} & k_{IS1} & k_{S01} \\ k_{SS2} & k_{II2} & k_{002} & k_{IS2} & k_{S02} & k_{I02} \\ k_{003} & k_{SS3} & k_{II3} & k_{S03} & k_{I03} & k_{IS3} \end{bmatrix}$$

$b_{gt} = 3 \times 1$ vector of unknown gyroscope fixed-drift trend parameters
$U_{gt} = 3 \times 6$ matrix of unknown gyroscope mass unbalance trend parameters
$\beta = 3 \times 1$ vector of vertical direction cosines (normalized gravity) $(\beta_1, \beta_2, \beta_3)^{\mathrm{T}}$
$\beta^1 = 6 \times 1$ vector with components $[\beta_1^2, \beta_2^2, \beta_3^2, \beta_1\beta_2, \beta_1\beta_3, \beta_2\beta_3]^{\mathrm{T}}$

$$x^g(t) = \begin{bmatrix} 1 \times 3 & 1 \times 9 & 1 \times 9 & 1 \times 15 & 1 \times 3 & 1 \times 3 & 1 \times 6 \\ b_g^{\mathrm{T}} & h_g^{1\mathrm{T}} & U_g^{1\mathrm{T}} & K_g^{1\mathrm{T}} & T_g^{\mathrm{T}} & b_{gt}^{\mathrm{T}} & U_{gt}^{1\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$$

The 48 unknown parameters will be modeled as random walks and random ramps (see Table 3.1) for the parameter identification problem to be discussed in Chapter 7 (Example 7.6).

3.7 MEAN AND COVARIANCE PROPAGATION

3.7.1 Propagating Gaussian Distributions

Linear stochastic systems are models for the evolution over time of the probability distributions of the state vector x of an RP. If the probability distribution of x is initially Gaussian, then linearity of the model preserves this property. A multivariate Gaussian distribution $\mathcal{N}(\hat{x}, P)$ is completely characterized by its vector mean

$$\hat{x} \stackrel{\text{def}}{=} \mathrm{E}\langle x \rangle \tag{3.105}$$

and its covarariance matrix (mean-squared deviation from the mean)

$$P \stackrel{\text{def}}{=} \mathrm{E}\big\langle [x - \hat{x}][x - \hat{x}]^{\mathrm{T}} \big\rangle. \tag{3.106}$$

Gaussianity of a probability distribution is a property preserved by linearity and the addition of constants because, if x is Gaussian, M is a matrix, and b is a vector, the

variate $y = Mx + b$ will have a Gaussian distribution with mean $\hat{y} = M\hat{x} + b$ and covariance $P_y = MPM^{\mathrm{T}}$. Therefore, the probability distribution of the state vector x can be effectively propagated over time by propagating just its mean $\hat{x}$ and covariance P.

The formulas derived here for propagating $\hat{x}$ and P are used in Kalman filtering for keeping track of the mean $\hat{x}$ (estimated state vector) and the covariance matrix P (mean-squared estimation error) between those epochs when measurements are used. The value of P is required for computing the Kalman filter gain matrix $\overline{K}$ to optimize the use of measurements. Formulas are derived in Chapter 4 for computing $\overline{K}$ and for updating $\hat{x}$ and P to include the effects of using measurements.

3.7.2 Propagation in Continuous Time

3.7.2.1 Propagation of the Mean For the linear stochastic system model

$$\dot{x}(t) = F(t)x(t) + G(t)w(t), \tag{3.107}$$

$$\mathrm{E}\langle w(t)\rangle = 0,$$

$$\mathrm{E}\langle w(t_1)w^{\mathrm{T}}(t_2)\rangle = Q(t_1)\delta(t_2 - t_1) \tag{3.108}$$

$$\delta(t_2 - t_1) = \begin{cases} 1, & t_1 = t_2 \\ 0, & t_1 \neq t_2, \end{cases}$$

one can equate the expected values of both sides of Equation 3.107 to obtain the equation for propagating the mean $\hat{x}$:

$$\mathrm{E}\langle \dot{x}(t)\rangle = \mathrm{E}\langle F(t)x(t) + G(t)w(t)\rangle \tag{3.109}$$

$$\frac{d}{dt}\hat{x}(t) = F(t)\hat{x}(t) + G(t)\mathrm{E}\langle w(t)\rangle \tag{3.110}$$

$$= F(t)\hat{x} + 0 \tag{3.111}$$

to obtain the general solution (derived in Chapter 2)

$$\hat{x}(t) = \Phi(t, t_0)\hat{x}(t_0) \tag{3.112}$$

$$\Phi(t, t_0) \stackrel{\mathrm{def}}{=} \exp\left(\int_{t_0}^{t} F(s)\, ds\right). \tag{3.113}$$

Given an initial value $\hat{x}(t_0)$, Equations 3.112 and 3.113 provide the general solution $\hat{x}(t)$ for all $t > t_0$.

3.7.2.2 Propagation of the Covariance Matrix Equation 3.112 propagates the mean $\hat{x}$ and Equation 3.107 propagates any x. Therefore, the deviation of x from the mean

$$\tilde{x}(t) \stackrel{\mathrm{def}}{=} x(t) - \hat{x}(t) \tag{3.114}$$

can be propagated by

$$\frac{d}{dt}\tilde{x}(t) = \frac{d}{dt}[x(t) - \hat{x}(t)] \tag{3.115}$$

$$= \frac{d}{dt}x(t) - \frac{d}{dt}\hat{x}(t) \tag{3.116}$$

$$= [F(t)x(t) + G(t)w(t)] - [F(t)\hat{x}(t)] \tag{3.117}$$

$$= F(t)\underbrace{[x(t) - \hat{x}(t)]}_{\tilde{x}} + G(t)w(t) \tag{3.118}$$

$$\frac{d}{dt}\tilde{x}(t) = F(t)\tilde{x}(t) + G(t)w(t). \tag{3.119}$$

The general forward solution to this linear differential equation with dependent variable $\tilde{x}$ (deviation from the mean) with the nonhomogeneous term $G(t)$ $w(t)$ is (from Chapter 2)

$$\tilde{x}(t) = \Phi(t, t_0)\tilde{x}(t_0) + \Phi(t, t_0)\int_{t_0}^{t} \Phi^{-1}(s, t_0)G(s)w(s)\,ds, \tag{3.120}$$

and its covariance is

$$P(t) \stackrel{\text{def}}{=} \mathrm{E}\langle \tilde{x}(t)\tilde{x}^{\mathrm{T}}(t)\rangle \tag{3.121}$$

$$\begin{aligned} &= \Phi(t, t_0)\mathrm{E}\langle \tilde{x}(t_0)\tilde{x}^{\mathrm{T}}(t_0)\rangle\Phi^{\mathrm{T}}(t, t_0) \\ &\quad + \Phi(t, t_0)\mathrm{E}\left\langle \left[\int_{t_0}^{t} \Phi^{-1}(s, t_0)G(s)w(s)\,ds\right] \right. \\ &\quad \left. \times \left[\int_{t_0}^{t} \Phi^{-1}(s, t_0)G(s)w(s)\,ds\right]^{\mathrm{T}}\right\rangle \Phi^{\mathrm{T}}(t, t_0) \\ &\quad + \text{terms } \xi \text{ for which } \mathrm{E}\langle \xi\rangle = 0. \end{aligned} \tag{3.122}$$

$$\begin{aligned} &= \Phi(t, t_0)P(t_0)\Phi^{\mathrm{T}}(t, t_0) \\ &\quad + \Phi(t, t_0)\left[\int_{t_0}^{t} \Phi^{-1}(s, t_0)G(s)Q(s)G^{\mathrm{T}}(s)\Phi^{-\mathrm{T}}(s, t_0)\,ds\right]\Phi^{\mathrm{T}}(t, t_0), \end{aligned} \tag{3.123}$$

where $\Phi(t, t_0)$ is defined by Equation, 3.113 and $\Phi(s, t_0)$ is defined by Equation 3.113 with $s = t$. The step between Equations 3.122 and 3.123 uses Equation 3.108 to collapse the product of two integrals into a single integral under the expectancy operator $\mathrm{E}\langle\cdot\rangle$.

3.7.2.3 Steady-State Equation Without Measurements Equation 3.123 is the general solution for forward propagation of the covariance matrix $P(t)$. It can be

differentiated to derive the equivalent differential equation and its steady-state form:

$$\dot{P} = FP + PF^{\mathrm{T}} + GQG^{\mathrm{T}}$$

$$O = FP + PF^{\mathrm{T}} + GQG^{\mathrm{T}} \quad \text{(steady-state)}.$$

The steady-state solution requires that F and Q be constant, and that the characteristic values of F reside in the left half of the complex plane.

3.7.3 Propagation in Discrete Time

Equations 3.112, 3.113, and 3.123 also provide a solution for propagation of means $\hat{x}$ and covariances P of the state variable x in discrete time. These are values

$$\hat{x}_k \stackrel{\text{def}}{=} \hat{x}(t_k) \tag{3.124}$$

$$P_k \stackrel{\text{def}}{=} P(t_k) \tag{3.125}$$

at discrete times $\{\cdots < t_{k-1} < t_k < t_{k+1} < \cdots\}$ with intersample intervals $\Delta t_k \stackrel{\text{def}}{=} t_k = t_{k-1}$.

The discrete-time analog of Equation 3.107 is

$$x_k = \Phi_k x_{k-1} + w_k \tag{3.126}$$

$$\Phi_k \stackrel{\text{def}}{=} \Phi(t_k, t_{k-1}), \tag{3.127}$$

where $\Phi(t_k, t_{k-1})$ is defined by Equation 3.113.

3.7.3.1 Propagation of the Mean Substituting $t = t_k$ and $t_0 = t_{k-1}$ into Equation 3.112 provides the solution as

$$\hat{x}_k = \Phi_k \hat{x}_{k-1}. \tag{3.128}$$

3.7.3.2 Propagating the Covariance Similarly, substituting $t = t_k$, $t_0 = t_{k-1}$ and Equation 3.127 into Equation 3.123 yields the solution as

$$P_k = \Phi_k P_{k-1} \Phi_k^{\mathrm{T}} + Q_{k-1}, \tag{3.129}$$

$$Q_{k-1} \stackrel{\text{def}}{=} \Phi(t_k, t_{k-1}) \left[\int_{t_{k-1}}^{t_k} \Phi^{-1}(s, t_{k-1}) Q(s) \Phi^{-\mathrm{T}}(s, t_{k-1})\, ds \right] \Phi^{\mathrm{T}}(t_k, t_{k-1}). \tag{3.130}$$

3.7.3.3 Time-Invariant Steady-State Equation without Measurements Equation 3.129 is the general solution for forward propagation of the covariance matrix P_k when no measurements are being used. In the case that Φ and Q are constant and the covariance matrix has a steady-state value $P_k = P_{k-1} = P_\infty$, the equation

for that steady-state value P_∞ will be

$$P_\infty = \Phi P_\infty \Phi^T + Q.$$

However, a steady-state solution P_∞ may not exist if any of the characteristic values of Φ are outside of the unit circle in the complex plane.

Example 3.9 (Steady-State Covariance of a Harmonic Resonator) The stochastic system model is

$$\dot{x}(t) = Fx(t) + w(t),$$
$$\mathrm{E}\langle w(t_1) w^{\mathrm{T}}(t_2)\rangle = \delta(t_1 - t_2) Q,$$
$$Q = \begin{bmatrix} 0 & 0 \\ 0 & q \end{bmatrix}$$

for an underdamped harmonic resonator driven by zero-mean white acceleration noise $w(t)$.

It is of interest to find whether the covariance of the process $x(t)$ reaches a finite *steady-state* value $P(\infty)$. Not every RP has a finite steady-state value, but it will turn out to be finite in this example.

Recall that the state-space model for the mass-spring harmonic resonator from Examples 2.2–2.3 and 2.6–2.7 has as its dynamic coefficient matrix the 2×2 constant matrix

$$F = \begin{bmatrix} 0 & 1 \\ -\dfrac{k_s}{m} & -\dfrac{k_d}{m} \end{bmatrix}$$
$$= \begin{bmatrix} 0 & 1 \\ -\omega_r^2 - \omega_d^2 & -2\omega_d \end{bmatrix},$$

where m is the supported mass, k_s is the spring elastic constant, and k_d is the dashpot damping coefficient. The alternate model parameters are

$$\omega_r = \sqrt{\frac{k_s}{m} - \frac{k_d^2}{4m^2}} \text{ (undamped resonant frequency)},$$
$$\tau_d = \frac{2m}{k_d} \text{ (damping time constant)},$$
$$\omega_d = \frac{1}{\tau_d} \text{ (damping frequency)}.$$

If the covariance matrix $P(t)$ reaches some state-state value $P(\infty)$ as $t \to +\infty$, the asymptotic covariance dynamic equation becomes

$$0 = \lim_{t \to \infty} \frac{d}{dt} P(t)$$
$$= FP(\infty) + P(\infty) F^{\mathrm{T}} + Q,$$

which is an *algebraic equation*. That is, it includes only algebraic operations (multiplies and adds) and no derivatives or integrals from the calculus. Moreover, it is a linear equation of the unknown elements p_{11}, $p_{12} = p_{21}$, p_{22} of the 2×2 symmetric matrix $P(\infty)$. Equation 3.131 is equivalent to three scalar linear equations:

$$0 = \sum_{k=1}^{2} f_{1k} p_{k1} + \sum_{k=1}^{2} p_{1k} f_{k1} + q_{11},$$

$$0 = \sum_{k=1}^{2} f_{1k} p_{k2} + \sum_{k=1}^{2} p_{1k} f_{k2} + q_{12},$$

$$0 = \sum_{k=1}^{2} f_{2k} p_{k2} + \sum_{k=1}^{2} p_{2k} f_{k2} + q_{22},$$

with known parameters

$$q_{11} = 0, q_{12} = 0, q_{22} = q$$

$$f_{11} = 0, f_{12} = 1, f_{21} = -\omega_r^2 - \omega_d^2, f_{22} = -2\omega_d.$$

The linear system of the equations above can be rearranged into the nonsingular 3×3 system of equations

$$\begin{bmatrix} 0 \\ 0 \\ q \end{bmatrix} = -\begin{bmatrix} 0 & 2 & 0 \\ -(\omega_r^2 + \tau_d^{-2}) & -\dfrac{2}{\tau_d} & 1 \\ 0 & -2(\omega_r^2 + \tau_d^{-2}) & -\dfrac{4}{\tau_d} \end{bmatrix} \begin{bmatrix} p_{11} \\ p_{12} \\ p_{22} \end{bmatrix}$$

with solution

$$\begin{bmatrix} p_{11} \\ p_{12} \\ p_{22} \end{bmatrix} = q \begin{bmatrix} \dfrac{\tau_d}{4(\omega_r^2 + \tau_d^{-2})} \\ 0 \\ \dfrac{\tau_d}{4} \end{bmatrix},$$

$$P(\infty) = \begin{bmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{bmatrix}$$

$$= \frac{q\tau_d}{4(\omega_r^2 + \tau_d^{-2})} \begin{bmatrix} 1 & 0 \\ 0 & (\omega_r^2 + \tau_d^{-2}) \end{bmatrix}.$$

Note that the steady-state state covariance depends linearly on the process noise covariance q. The steady-state covariance of velocity also depends linearly on the damping time constant τ_d. The dimensionless quantity $2\pi\omega_r\tau_d$ is called the *quality factor*, *Q-factor*, or simply *the Q* of a resonator. It equals the number of cycles that the unforced resonator will go through before its amplitude falls to $1/e \approx 37\%$ of its initial amplitude.

3.8 RELATIONSHIPS BETWEEN MODEL PARAMETERS

3.8.1 Parameters of Continuous and Discrete Models

The mathematical forms of the applications models used in Kalman filtering (in discrete time) and Kalman–Bucy filtering (in continuous time) are summarized in Table 3.3. All these models are written in general-purpose mathematical forms, with the dimensions and values of the model parameters F, Q, H, R, and Φ, depending on the application. The relationship between Φ and F (shown in the table) is derived in Chapter 2, and the solutions for nonhomogeneous differential equations derived in Chapter 2 were used—with a little stochastic calculus—to derive the relationship between the parameters $Q(t)$ and Q_{k-1} in Equation 3.130.

Sensor Models The last column of Table 3.3 includes a new type of model, labeled "Output Model" at the top of the table. This type of model was introduced in Section 2.3.4 of Chapter 2. Output models, which are for the sensors used in estimating the state vector x of a Kalman filter, contain the parameters H and R. The relationship between $H(t)$ and H_k is shown in the table.

The relationship between $R(t)$ and R_k depends on the type of filtering used before sampling the measurements z_k at discrete times t_k. These types of relationships depend on the same stochastic calculus used in deriving the relationships between $Q(t)$ and Q_{k-1}. In Section 3.8.3, we will derive the $R(t) \rightarrow R_k$ relationship for some common types of anti-aliasing filters.

The Utility of Continuous-Time Models These relationsips are important for many applications of Kalman filtering where the source models are in continuous time, and the Kalman filter must be implemented on a computer in discrete time. Engineers and scientists often find it more natural and reliable to start with the model in continuous time and then convert the model to discrete time after they have full confidence in the model they have developed. The risk of blunders in filter derivation can usually be reduced by following the maxim:

> Think continuously. Act discretely.

TABLE 3.3 Parameters of Stochastic System Models

Filter Type	Dynamic Model	Output Model
Kalman–Bucy	$\dot{x}(t) = F(t)\,x(t) + w(t)$	$z(t) = H(t)\,x(t) + v(t)$
Parameters:	$F(t) = \dfrac{\partial \dot{x}}{\partial x}$	$H\,(t) = \dfrac{\partial z}{\partial x}$
	$Q(t) = \mathrm{E}\,\langle\, w(t)w^{\mathrm{T}}\,(t)\rangle$	$R(t) = \mathrm{E}\,\langle v(t)v^{\mathrm{T}}(t)\rangle$
Kalman	$x_k = \Phi_{k-1}\,x_{k-1} + w_{k-1}$	$z_k = H_k\,x_k + v_k$
Parameters:	$\Phi_{k-1} = \exp\,(\int_{t_{k-1}}^{t_k} F(s)\,\mathrm{ds})$	$H_k = H(t_k)$
	$Q_{k-1} = \mathrm{E}\langle w_{k-1}w_{k-1}^{\mathrm{T}}\rangle$	$R_k = \mathrm{E}\,\langle v_k v_k^{\mathrm{T}}\rangle$

3.8.2 Relationship between $Q(t)$ and Q_{k-1}

The solution has been derived as Equation 3.130, which we reproduce here as

$$Q_{k-1} = \Phi(t_k, t_{k-1}) \left[\int_{t_{k-1}}^{t_k} \Phi^{-1}(s, t_{k-1}) Q(s) \Phi^{-\mathrm{T}}(s, t_{k-1})\, ds \right] \Phi^{\mathrm{T}}(t_k, t_{k-1}). \quad (3.131)$$

Solutions for some time-invariant system models are shown in Table 3.4.

Covariance Units Note that the process noise covariance matrices $Q(s)$ and Q_{k-1} have different physical units. The units of $w(t) \propto \dot{x}$ are the units of the state vector x divided by time, and the units of $w_{k-1} \propto x$ are the same as those of the state vector x. One might then expect that the units of $Q(s) = \mathrm{E}\langle w(s) w^{\mathrm{T}}(s) \rangle$ would be the units of xx^{T} divided by time squared. However, the expectancy operator combined with the stochastic integral (of Itô or Stratonovich) makes the units of $Q(s)$ be the units of xx^{T} divided by time—a potential source of confusion. Integration of $Q(s)$ over time then makes the units of Q_k equal to those of xx^{T}.

Example 3.10 (Q_{k-1} for the Harmonic Resonator Model) Consider the problem of determining the covariance matrix Q_{k-1} for the equivalent discrete-time model

$$x_k = \Phi_{k-1} x_{k-1} + w_{k-1},$$
$$\mathrm{E}\langle w_{k-1} w_{k-1}^{\mathrm{T}} \rangle = Q_{k-1}$$

for a harmonic resonator driven by white acceleration noise, given the variance q of

TABLE 3.4 Equivalent Constant Parameters, Continuous- and Discrete-time Models

Continuous Time		Discrete Time	
F	Q	Φ_k	Q_k
$[0]$	$[q]$	$[1]$	$[q\,\Delta\, t]$
$[-1/\tau]$	$[q]$	$[\exp(-\Delta t/\tau)]$	$\left[\frac{q\tau[1-\exp(-2\,\Delta t/\tau)]}{2}\right]$
$\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 0 & q \end{bmatrix}$	$\begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}$	$q \begin{bmatrix} \frac{(\Delta t)^3}{3} & \frac{(\Delta t)^2}{2} \\ \frac{(\Delta t)^2}{2} & \Delta t \end{bmatrix}$
$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & q \end{bmatrix}$	$\begin{bmatrix} 1 & \Delta t & \frac{(\Delta t)^2}{2} \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix}$	$q \begin{bmatrix} \frac{(\Delta t)^5}{20} & \frac{(\Delta t)^4}{8} & \frac{(\Delta t)^3}{6} \\ \frac{(\Delta t)^4}{8} & \frac{(\Delta t)^3}{6} & \frac{(\Delta t)^2}{2} \\ \frac{(\Delta t)^3}{6} & \frac{(\Delta t)^2}{2} & \Delta t \end{bmatrix}$

the process noise in its continuous-time model

$$\frac{d}{dt}x(t) = Fx(t) + w(t),$$
$$\mathrm{E}\langle w(t_1)w^{\mathrm{T}}(t_2)\rangle = \delta(t_1 - t_2)Q,$$
$$Q = \begin{bmatrix} 0 & 0 \\ 0 & q \end{bmatrix},$$

where ω is the resonant frequency, τ is the damping time constant, ξ is the corresponding damping "frequency" (i.e., has frequency units), and q is the process noise covariance in continuous time. [This stochastic system model for a harmonic resonator driven by white acceleration noise $w(t)$ is derived in Examples 3.4 and 3.9.]

Following the derivation of Example 2.6, the fundamental solution matrix for the *unforced* dynamic system model can be expressed in the form

$$\Phi(t) = e^{Ft}$$
$$= e^{-t/\tau}\begin{bmatrix} \dfrac{S(t) + C(t)\omega\tau}{\omega\tau} & \dfrac{S(t)}{\omega} \\ -\dfrac{S(t)(1+\omega^2\tau^2)}{\omega\tau^2} & \dfrac{-S(t) + C(t)\omega\tau}{\omega\tau} \end{bmatrix},$$
$$S(t) = \sin(\omega t),$$
$$C(t) = \cos(\omega t)$$

in terms of its resonant frequency ω and damping time-constant τ. Its matrix inverse is

$$\Phi^{-1}(s) = \frac{e^{s/\tau}}{\omega\tau^2}\begin{bmatrix} \tau[\omega\tau C(s) - S(s)] & -\tau^2 S(s) \\ (1+\omega^2\tau^2)S(s) & \tau[\omega\tau C(s) + S(s)] \end{bmatrix}$$

at time $t = s$. Consequently, the indefinite integral matrix is

$$\Psi(t)$$
$$= \int_0^t \Phi^{-1}(s)\begin{bmatrix} 0 & 0 \\ 0 & q \end{bmatrix}\Phi^{\mathrm{T}-1}(s)ds$$
$$= \frac{q}{\omega^2\tau^2}\int_0^t \begin{bmatrix} \tau^2 S(s)^2 & -\tau S(s)[\omega\tau C(s)+S(s)] \\ -\tau S(s)[\omega\tau C(s)+S(s)] & [\omega\tau C(s)+S(s)]^2 \end{bmatrix} e^{2s/\tau}ds$$
$$= \begin{bmatrix} \dfrac{q\tau\{-\omega^2\tau^2+[2S(t)^2-2C(t)\omega S(t)\tau+\omega^2\tau^2]\zeta^2\}}{4\omega^2(1+\omega^2\tau^2)} & \dfrac{-qS(t)^2\zeta^2}{2\omega^2} \\ \dfrac{-qS(t)^2\zeta^2}{2\omega^2} & \dfrac{q\{-\omega^2\tau^2+[2S(t)^2+2C(t)\omega S(t)\tau+\omega^2\tau^2]\zeta^2\}}{4\omega^2\tau} \end{bmatrix},$$
$$\zeta = e^{t/\tau}.$$

The discrete-time covariance matrix Q_{k-1} from Equation 3.130 is then

$$
\begin{aligned}
Q_{k-1} &= \Phi(\Delta t)\Psi(\Delta t)\Phi^{\mathrm{T}}(\Delta t) \\
&= \begin{bmatrix} q_{11} & q_{12} \\ q_{21} & q_{22} \end{bmatrix}, \\
q_{11} &= \frac{q\tau\{\omega^2\tau^2(1-e^{-2\Delta t/\tau}) - 2S(\Delta t)e^{-2\Delta t/\tau}[S(\Delta t)+\omega\tau C(\Delta t)]\}}{4\omega^2(1+\omega^2\tau^2)}, \\
q_{12} &= \frac{qe^{-2\Delta t/\tau}S(\Delta t)^2}{2\omega^2}, \\
q_{21} &= q_{12}, \\
q_{22} &= \frac{q\{\omega^2\tau^2(1-e^{-2\Delta t/\tau}) - 2S(\Delta t)e^{-2\Delta t/\tau}[S(\Delta t)-\omega\tau C(\Delta t)]\}}{4\omega^2\tau}.
\end{aligned}
$$

Note that the structure of the discrete-time process noise covariance Q_{k-1} for this example is quite different from that of the continuous-time process noise Q. In particular, Q_{k-1} is a full matrix, although Q is a sparse matrix.

3.8.2.1 First-Order Approximation of Q_k for Constant F and G The justification of a truncated power series expansion for Q_k when F and G are constant is as follows:

$$
Q_k = \sum_{i=1}^{\infty} \frac{Q^i \Delta t^i}{i!} \tag{3.132}
$$

Consider the Taylor series expansion of Q_k about t_{k-1}, where

$$
\begin{aligned}
Q^i &= \left.\frac{d^i Q}{dt^i}\right|_{t=t_{k-1}}, \\
\dot{Q} &= FQ_k + Q_k F^{\mathrm{T}} + GQ(t)G^{\mathrm{T}}, \\
Q^{(1)} &= \dot{Q}(t_{k-1}) = GQ(t)G^{\mathrm{T}} \text{since} Q(t_{k-1}) = 0, \\
Q^{(2)} &= \ddot{Q}(t_{k-1}) = F\dot{Q}(t_{k-1}) + \dot{Q}(t_{k-1})F^{\mathrm{T}}, \\
&= FQ^{(1)} + Q^{(1)}F^{\mathrm{T}}, \\
&\vdots \\
Q^{(i)} &= FQ^{(i-1)} + Q^{(i-1)}F^{\mathrm{T}}, \quad i = 1, 2, 3, \ldots.
\end{aligned}
$$

Taking only first-order terms in the above series,

$$Q_k \approx GQ(t)G^{\mathrm{T}}\Delta t. \tag{3.133}$$

This is not always a good approximation, as is shown in the following example.

Example 3.11 (First-Order Approximation of Q_k for the Harmonic Resonator) Let us see what happens if this first-order approximation

$$Q_k \approx Q\Delta t$$

is applied to the previous example of the harmonic resonator with acceleration noise.

The solution to the steady-state "state covariance" equation (i.e., the equation of covariance of the state vector itself, not the estimation error)

$$P_\infty = \Phi P_\infty \Phi^{\mathrm{T}} + Q\Delta t$$

has the solution (for $\theta = 2\ \pi f_{\text{resonance}}/f_{\text{sampling}}$)

$$\begin{aligned}\{P_\infty\}_{11} &= q\Delta t e^{-2\Delta t/\tau}\sin(\theta)^2(e^{-2\Delta t/\tau}+1)/D,\\ D &= \omega^2(e^{-2\Delta t/\tau}-1)\\ &\quad\times(e^{-2\Delta t/\tau}-2e^{-2\Delta t/\tau}\cos(\theta)+1)(e^{-2\Delta t/\tau}+2e^{-2\Delta t/\tau}\cos(\theta)+1)\end{aligned}$$

for its upper-left element, which is the steady-state mean-squared resonator displacement. Note, however, that

$$\{P_\infty\}_{11} = 0 \quad \text{if} \quad \sin(\theta) = 0,$$

which would imply that there is no displacement if the sampling frequency is twice the resonant frequency. This is absurd, of course. This proves by contradiction that

$$Q_k \neq Q\Delta t$$

in general—even though it may be a reasonable approximation in some instances.

Example 3.12 (Q_k versus Q for Exponentially Correlated Noise Models) The exponentially correlated noise model is linear and time-invariant, and the correspondence between the discrete-time Q_k and the analogous Q_c in continuous time can be

derived in closed form. Starting with the continuous-time model

$$\dot{x} = \frac{-1}{\tau}x(t) + w(t), \text{ for } w(t) \in \mathcal{N}(0,\, Q_c),$$

$$F = \frac{-1}{\tau}$$

$$\Phi_k = \Phi(\Delta t) = \exp\left(\int_0^{\Delta t} F\, ds\right) = \exp(-\Delta t/\tau)$$

$$Q_k = \Phi_k^2 \int_0^{\Delta t} \Phi^{-1}(s) Q_c \Phi^{-T}(s)\, ds = \frac{\tau}{2}[1 - \exp(-2\,\Delta t/\tau)]Q_c$$

$$Q_c = \frac{2}{\tau[1 - \exp(-2\,\Delta t/\tau)]} Q_k.$$

3.8.2.2 Van Loan's Method for Computing Q_k from Continuous Q For the general linear time-invariant model in continuous time

$$\dot{x}(t) = Fx(t) + Gw(t) \text{ with } w(t) \in \mathcal{N}(0, Q_c),$$

and with n state variables, form the $2n \times 2n$ matrix

$$M = \Delta t \begin{bmatrix} -F & GQ_cG^{\mathrm{T}} \\ 0 & F^{\mathrm{T}} \end{bmatrix}, \tag{3.134}$$

partitioned into a 2×2 array of $n \times n$ blocks.

A 1978 article by Charles F. Van Loan [276] shows how the $2n \times 2n$ matrix exponential $\exp(M)$, which can be computed numerically using the MATLAB function `expm`, can also be partitioned into a 2×2 array of $n \times n$ blocks, where

$$\exp(M) = \begin{bmatrix} \Psi & \Phi_k^{-1} Q_k \\ 0 & \Phi_k^{\mathrm{T}} \end{bmatrix}, \tag{3.135}$$

and the equivalent discrete-time model parameters

$$\texttt{Phik} \stackrel{\text{def}}{=} \Phi_k$$
$$\texttt{Qk} \stackrel{\text{def}}{=} Q_k$$

can be computed in MATLAB by the script

```
   M = DeltaT*[-F,G*Qc*G';zeros(n),F'];
   N = expm(M);
Phik = N(n+1:2*n,n+1:2*n)';
  Qk = Phik*N(1:n,n+1:2*n);
```

where the MATLAB variable `Qc` $\stackrel{\text{def}}{=} Q_c$.

Van Loan's method is coded in the MATLAB function `VanLoan` on the companion CD.

Example 3.13 (Using Van Loan's Method) Let the continuous time model for a linear time-invariant stochastic system have parameter values

$$F = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad GQ_cG^{\mathrm{T}} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

and its equivalent implementation in discrete time require time-step $\Delta t = 1$. Then

$$M = \Delta t \begin{bmatrix} -F & GQ_cG^{\mathrm{T}} \\ 0 & F^{\mathrm{T}} \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$\begin{aligned} \exp(M) &= I + M + \frac{1}{2!}M^2 + \frac{1}{3!}M^3 + \frac{1}{4!}M^4 + \cdots \\ &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} + \frac{1}{2}\begin{bmatrix} 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \\ &\quad + \frac{1}{6}\begin{bmatrix} 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} + \frac{1}{24}\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} + \text{more zero terms} \\ &= \begin{bmatrix} 1 & -1 & -\frac{1}{6} & -\frac{1}{2} \\ 0 & 1 & \frac{1}{2} & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} \Psi & \Phi_k^{-1}Q_k \\ 0 & \Phi_k^{\mathrm{T}} \end{bmatrix}, \end{aligned}$$

from which

$$\Phi_k = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$$

$$Q_k = \Phi_k\{\Phi_k^{-1} Q_k\} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \left\{ \begin{bmatrix} -\frac{1}{6} & -\frac{1}{2} \\ \frac{1}{2} & 1 \end{bmatrix} \right\} = \begin{bmatrix} \frac{1}{3} & \frac{1}{2} \\ \frac{1}{2} & 1 \end{bmatrix}.$$

Because $F^2 = 0$, the value of Φ_k obtained above can easily be verified by calculating it as

$$\begin{aligned} \Phi_k &= \exp(\Delta t\, F) \\ &= I + \Delta t\, F + \frac{1}{2!}\underbrace{(\Delta t\, F)^2}_{=0} + \cdots \\ &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \end{aligned}$$

which can be verified in MATLAB as well:

```
>> F = [0,1;0,0],
F =
     0     1
     0     0
>> Qc = [0,0;0,1],
Qc =
     0     0
     0     1
>> DeltaT = 1;
>> M = DeltaT*[-F, Qc;zeros(2),F'],
M =
     0    -1     0     0
     0     0     0     1
     0     0     0     0
     0     0     1     0
```

```
>> N = expm(M),
N =
    1.0000 -1.0000 -0.1667 -0.5000
         0  1.0000  0.5000  1.0000
         0       0  1.0000       0
         0       0  1.0000  1.0000
>> Phi = N(3:4, 3:4)',
Phi =
     1      1
     0      1
>> Qk = Phi*N(1:2,3:4),
Qk =
    0.3333    0.5000
    0.5000    1.0000
```

3.8.3 Relationship between $R(t)$ and R_k

Sensor Noise Calibration In practice, $R(t)$ and/or R_k should be determined by measuring the actual sensor outputs with constant inputs—as part of sensor characterization and/or calibration. In either case (continuous time or discrete time), the

noise measurements would also be used to determine whether the appropriate noise model is zero-mean Gaussian white noise. If not, then the same noise data (or their PSD) can be used to devise a suitable shaping filter (i.e., an auxiliary noise model as a stochastic system with Gaussian white noise inputs). The values of $R(t)$ or R_k would then be those which approximate—at the output of the shaping filter—the statistical properties of the measured sensor noise.

Sensor Output Bias This sort of noise calibration is also needed to detect and correct any nonzero mean of the noise, which is usually called *sensor output bias*. Sensor output bias can also be estimated and compensated for by making it a system state variable, to be estimated by a Kalman filter. This approach also works for biases that are not constant.

3.8.3.1 Integrating Sensors Even if $R(t)$ and R_k are already zero-mean Gaussian white noise, the relationship between them depends upon the way that the discrete-time sensor processes internal continuous-time noise. If it is an integrating sensor or uses an integrate-and-hold circuit before sampling, then

$$v_k = \int_{t_{k-1}}^{t_k} v(t)\,dt, \tag{3.136}$$

$$R_k = \bar{R} \tag{3.137}$$

$$= \frac{1}{t_k - t_{k-1}} \int_{t_{k-1}}^{t_k} R(t)\,dt, \tag{3.138}$$

where $\overline{R}$ is the time-averaged value of $R(t)$ over the interval $t_{k-1} < t \leq t_k$. If the analog noise $v(t)$ is zero-mean and white, then the integrate-and-hold circuit is the ideal low-pass anti-aliasing filter. It integrates only over the intersampling interval, which keeps the output noise white (i.e., introduces no time correlation).

3.8.3.2 Other Anti-Alias Filters

Tapped Delay Lines Filters implemented with tapped delay lines can also keep total delays with an intersampling interval by keeping the total delay less than or equal to the output intersampling interval. The internal sampling rate can still be much higher than the rate at which the output is sampled.

IIR Filters Analog RC filters, on the other hand, have infinite impulse response (IIR). This introduces some lag beyond the intersampling interval, which creates time correlation in the output noise—even if the input noise is white.

Calculating R_k In general, the filtered output noise covariance R_k can be calculated by using the input noise model and the transfer function of the anti-aliasing filter.

3.9 ORTHOGONALITY PRINCIPLE

3.9.1 Estimators Minimizing Expected Quadratic Loss Functions

A block diagram representation of an estimator of the state of a system represented by Equation 3.82 is shown in Figure 3.9. The estimate $\hat{x}(t)$ of $x(t)$ will be the output of a Kalman filter.

The *estimation error* is defined as the difference between the *true* value of a random variable $x(t)$ and an *estimate* $\hat{x}(t)$.

A quadratic loss function of the estimation error has the form

$$[x(t) - \hat{x}(t)]^{\mathrm{T}} M[x(t) - \hat{x}(t)], \tag{3.139}$$

where M is an $n \times n$ symmetric, positive-definite *weighting matrix.*

An *optimal* estimator for a particular quadratic loss function is defined as that estimate $\hat{x}(t)$ minimizing the *expected value* of the loss, with the probabilities conditioned on the observation $z(t)$. It will be shown that the optimal estimate of $x(t)$ (minimizing the average of a quadratic cost function) is the conditional expectation of $x(t)$ given the observation $z(t)$:

$$\hat{x} = \mathrm{E}\langle x(t)|z(t)\rangle \quad \text{minimizes} \quad \mathrm{E}\langle [x(t) - \hat{x}(t)]^{\mathrm{T}} M[x(t) - \hat{x}(t)]|z(t)\rangle. \tag{3.140}$$

Let $z(t)$, $0 \le t \le t_1$ be the observed quantity, and it is desired to estimate $x(t)$ at $t = t_2$. Then Equation 3.140 assumes the form

$$\hat{x}(t_2) = \mathrm{E}\langle x(t_2)|z(t), 0 \le t \le t_1\rangle \tag{3.141}$$

and the corresponding equation for a similar discrete model is

$$\hat{x}_{k_2} = \mathrm{E}\langle \hat{x}_{k_2}|z_1, z_2, \ldots, z_{k_1}\rangle, \quad 1 \le k_2 \le k_1. \tag{3.142}$$

Let

$$J = \mathrm{E}\langle [x(t) - \hat{x}(t)]^{\mathrm{T}} M[x(t) - \hat{x}(t)]|z(t)\rangle. \tag{3.143}$$

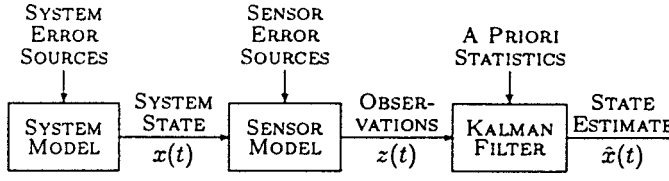


Figure 3.9 Block diagram of estimator model.

Recall that $\hat{x}(t)$ is a nonrandom function of the observations

$$0 = \frac{dJ}{d\hat{x}} \tag{3.144}$$

$$= -2M\mathrm{E}\langle[x(t) - \hat{x}(t)]|z(t)\rangle, \tag{3.145}$$

$$\mathrm{E}\langle\hat{x}(t)|z(t)\rangle = \hat{x}(t) = \mathrm{E}\langle x(t)|z(t)\rangle. \tag{3.146}$$

This proves the result of Equation 3.140. If $x(t)$ and $z(t)$ are jointly normal (Gaussian), the nonlinear minimum variance and linear minimum variance estimators coincide:

$$\mathrm{E}\langle x_{k_2}|z_1, z_2, \ldots, z_{k_1}\rangle = \sum_{i=1}^{k_1} \alpha_i z_i \tag{3.147}$$

and

$$\mathrm{E}\langle x(t_2)|z(t), 0 \le t \le t_1\rangle = \int_0^{t_1} \alpha(t, \tau)z(\tau)\, d\tau. \tag{3.148}$$

Proof for the Discrete Case Recall the properties of jointly Gaussian processes from Section 3.2.2. Let the probability density

$$p[x_{k_2}|z_{k_1}] \tag{3.149}$$

be Gaussian and let $\alpha_1, \alpha_2, \ldots, \alpha_{k1}$ satisfy

$$\mathrm{E}\left\langle \left[x_{k_2} - \sum_{i=1}^{k_1} \alpha_i z_i \right] z_j^{\mathrm{T}} \right\rangle = 0, \quad j = 1, \ldots, k_1 \tag{3.150}$$

and

$$k_1 < k_2,\ k_1 = k_2, \quad \text{or} \quad k_1 > k_2. \tag{3.151}$$

The existence of vectors α_i satisfying this equation is guaranteed because the covariance $[z_i, z_j]$ is nonsingular.

The vectors

$$\left[x_{k_2} - \sum \alpha_i z_i \right] \tag{3.152}$$

and z_i are independent. Then it follows from the zero-mean property of the sequence x_k that

$$\mathrm{E}\left\langle \left[x_{k_2} - \sum_{i=1}^{k_1} \alpha_i z_i \right] \middle| z_1, \ldots, z_{k_1} \right\rangle = \mathrm{E}\langle x_{k_2} - \sum_{i+1}^{k_1} \alpha_i z_i \rangle$$

$$= 0,$$

$$\mathrm{E}\langle x_{k_2} | z_1, z_2, \ldots, z_{k_1} \rangle = \sum_{i=1}^{k_1} \alpha_i z_i.$$

The proof of the continuous case is similar.

The linear minimum variance estimator is unbiased, that is,

$$\mathrm{E}\langle x(t) - \hat{x}(t) \rangle = 0, \tag{3.153}$$

where

$$\hat{x}(t) = \mathrm{E}\langle x(t) | z(t) \rangle. \tag{3.154}$$

In other words, an unbiased estimate is one whose expected value is the same as that of the quantity being estimated.

3.9.2 Orthogonality Principle

The nonlinear solution $\mathrm{E}\langle x | z \rangle$ of the estimation problem is not simple to evaluate. If x and z are jointly normal, then $\mathrm{E}\langle x|z \rangle = \alpha_1 z + \alpha_0$.

Let x and z be scalars, and let M be a 1×1 weighting matrix. The constants α_0 and α_1 that minimize the *mean-squared error*

$$e = \mathrm{E}\langle [x - (\alpha_0 + \alpha_1 z)]^2 \rangle = \int_{\infty}^{\infty} \int_{\infty}^{\infty} [x - (\alpha_0 + \alpha_1 z)]^2 p(x, z)\, dx\, dz \tag{3.155}$$

are given by

$$\alpha_1 = \frac{r\sigma_x}{\sigma_z},$$

$$\alpha_0 = \mathrm{E}\langle x \rangle - \alpha_1 \mathrm{E}\langle z \rangle,$$

and the resulting minimum mean-squared error $e_{\min}$ is

$$e_{\min} = \sigma_x^2 (1 - r^2), \tag{3.156}$$

where the ratio

$$r = \frac{\mathrm{E}\langle xz \rangle}{\sigma_x \sigma_z} \tag{3.157}$$

is called the *correlation coefficient* of x and z, and σ_x, σ_z are standard deviations of x and z, respectively.

Suppose α_1 is specified. Then

$$\frac{d}{d\alpha_0}\mathrm{E}\langle[x-\alpha_0-\alpha_1 z]^2\rangle = 0 \tag{3.158}$$

and

$$\alpha_0 = \mathrm{E}\langle x\rangle - \alpha_1 \mathrm{E}\langle z\rangle. \tag{3.159}$$

Substituting the value of α_0 in $\mathrm{E}\langle[x-\alpha_0-\alpha_1 z]^2\rangle$ yields

$$\begin{aligned}\mathrm{E}\langle[x-\alpha_0-\alpha_1 z]^2\rangle &= \mathrm{E}\langle[x-\mathrm{E}\langle x\rangle-\alpha_1(z-\mathrm{E}\langle z\rangle)]^2\rangle\\ &= \mathrm{E}\langle[(x-\mathrm{E}\langle x\rangle)-\alpha_1(z-\mathrm{E}\langle z\rangle)]^2\rangle\\ &= \mathrm{E}\langle[x-\mathrm{E}\langle x\rangle]^2\rangle+\alpha_1^2\mathrm{E}\langle[z-\mathrm{E}\langle z\rangle]^2\rangle\\ &\quad -2\alpha_1\mathrm{E}\langle(x-\mathrm{E}\langle x\rangle)(z-\mathrm{E}\langle z\rangle)\rangle,\end{aligned}$$

and differentiating with respect to α_1 as

$$\begin{aligned}0 &= \frac{d}{d\alpha_1}\mathrm{E}\langle[x-\alpha_0-\alpha_1 z]^2\rangle\\ &= 2\alpha_1\mathrm{E}\langle(z-\mathrm{E}\langle z\rangle)^2\rangle - 2\mathrm{E}\langle(x-\mathrm{E}\langle x\rangle)(z-\mathrm{E}\langle z\rangle)\rangle, \qquad (3.160)\\ \alpha_1 &= \frac{\mathrm{E}\langle(x-\mathrm{E}\langle x\rangle)(z-\mathrm{E}\langle z\rangle)\rangle}{\mathrm{E}\langle(z-\mathrm{E}\langle z\rangle)^2\rangle}\\ &= \frac{r\sigma_x\sigma_z}{\sigma_z^2}\\ &= \frac{r\sigma_x}{\sigma_z}, \qquad (3.161)\\ e_{\min} &= \sigma_x^2 - 2r^2\sigma_x^2 + r^2\sigma_x^2\\ &= \sigma_x^2(1-r^2).\end{aligned}$$

Note that, if one assumes that x and z have zero means,

$$\mathrm{E}\langle x\rangle = \mathrm{E}\langle z\rangle = 0; \tag{3.162}$$

then we have the solution

$$\alpha_0 = 0. \tag{3.163}$$

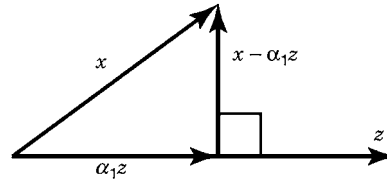


Figure 3.10 Orthogonality diagram.

Orthogonality Principle The constant α_1 that minimizes the mean-squared error

$$e = \mathrm{E}\langle [x - \alpha_1 z]^2 \rangle \tag{3.164}$$

is such that $x - \alpha_1 z$ is *orthogonal* to z. That is,

$$\mathrm{E}\langle [x - \alpha_1 z] z \rangle = 0, \tag{3.165}$$

and the value of the minimum mean-squared error is given by the formula

$$e_m = \mathrm{E}\langle (x - \alpha_1 z) x \rangle. \tag{3.166}$$

3.9.3 A Geometric Interpretation of Orthogonality

Consider all random variables as vectors in abstract vector spaces. The inner product of x and z is taken as the second moment $\mathrm{E}\langle xz \rangle$. Thus,

$$\mathrm{E}\langle x^2 \rangle = \mathrm{E}\langle x^{\mathrm{T}} x \rangle \tag{3.167}$$

is the square of the length of x. The vectors x, z, $\alpha_1 z$, and $x - \alpha_1 z$ are as shown in Figure 3.10.

The mean-squared error $\mathrm{E}\langle (x - \alpha_1 z)^2 \rangle$ is the square of the length of $x - \alpha_1 z$. This length is minimum if $x - \alpha_1 z$ is orthogonal (perpendicular) to z,

$$\mathrm{E}\langle (x - \alpha_1 z) z \rangle = 0. \tag{3.168}$$

We will apply the orthogonality principle to derive Kalman estimators in Chapter 4.

3.10 SUMMARY

3.10.1 Important Points to Remember

Probabilities Are Measures That is, they are functions whose arguments are *sets* of points, not individual points. The *domain* of a probability measure P is a *sigma*

algebra of subsets of a given set S, called the *measurable sets* of S. The sigma algebra of measurable sets has an algebraic structure under the operations of *set union* and *set complement*. The measurable sets always include the empty set $\{\ \}$ and the set S, and the probability $P(S) = 1$, $P(\{\ \}) = 0$, $P(A \cup B) + P(A \cap B) = P(A) + P(B)$ for all measurable sets A and B. A *probability space* is characterized by a set S, a sigma algebra of its measurable subsets, and a probability measure defined on the measurable subsets.

Events Form a Sigma Algebra of Outcomes of an Experiment A *statistical experiment* is an undertaking with an uncertain outcome. The set of *all* possible outcomes of an experiment is called a *sample space*. An event is said to *occur* if the outcome of an experiment is an element of the event.

Independent Events A collection of events is called *mutually independent* if the occurrence or nonoccurrence of any finite number of them has no influence on the possibilities for occurrence or nonoccurrence of the others.

RVs Are Functions A *scalar RV* is a real-valued function defined on the sample space of a probability space such that, for every open interval (a, b), $-\infty < a \leq b < +\infty$, the set

$$f^{-1}((a, b)) = \{s \in S | a < f(s) < b\}$$

is an event (i.e., is in the sigma algebra of events). A vector-valued RV has scalar RVs as its components. An RV is also called a *variate*.

Random processes (RPs) are functions of time with RVs as their values. A *process* is the evolution over time of a system. If the future state of the system can be predicted from its initial state and its inputs, then the process is considered *deterministic*. Otherwise, it is called *nondeterministic*. If the possible states of a nondeterministic system at any time can be represented by an RV, then the evolution of the state of the system is an RP, or a *stochastic process*. Formally, an RP or a stochastic process is a function f defined on a time interval with RVs as its values $f(t)$.

An RP is called:

- A *Bernoulli process*, or *independent, identically distributed (i.i.d.)* process if the probability distribution of its values at any time is independent of its values at any other time.
- A *Markov process* if, for any time t, the probability distribution of its state at any time $\tau > t$, given its state at time t, is the same as its probability distribution given its state at all times $s \leq t$.
- A *Gaussian process* if the probability distribution of its possible values at any time is a Gaussian distribution.
- *Stationary* if certain statistics of its probability distributions are invariant under shifts of the time origin. If only its first and second moments are invariant, it

is called *wide-sense stationary* (WSS) or *weak-sense stationary*. If all its statistics are invariant, it is called *strict sense stationary*.

Ergodic if the probability distribution of its values at any one time, over the ensemble of sample functions, equals the probability distribution over all time of the values of randomly chosen member functions.

Orthogonal to another random process if the expected value of their pointwise product is zero.

3.10.2 Important Equations to Remember

The density function of an n-vector-valued (or *multivariate*) *Gaussian probability distribution* $\mathcal{N}(\bar{x}, P)$ has the functional form

$$p(x) = \frac{1}{\sqrt{(2\pi)^n \det P}} e^{-(1/2)(x-\bar{x})^{\mathrm{T}} P^{-1}(x-\bar{x})},$$

where $\bar{x}$ is the *mean* of the distribution and P is the covariance matrix of deviations from the mean.

A *linear stochastic process in continuous time* with state x and state covariance P has the model equations

$$\begin{aligned}
\dot{x}(t) &= F(t)x(t) + G(t)w(t), \\
z(t) &= H(t)x(t) + v(t), \\
\dot{P}(t) &= F(t)P(t) + P(t)F^{\mathrm{T}}(t) + G(t)Q(t)G^{\mathrm{T}}(t),
\end{aligned}$$

where $Q(t)$ is the covariance of zero-mean *plant noise* $w(t)$. A *discrete-time linear stochastic process* has the model equations

$$\begin{aligned}
x_k &= \Phi_{k-1}x_{k-1} + G_{k-1}w_{k-1}, \\
z_k &= H_k x_k + v_k, \\
P_k &= \Phi_{k-1}P_{k-1}\Phi_{k-1}^{\mathrm{T}} + G_{k-1}Q_{k-1}G_{k-1}^{\mathrm{T}},
\end{aligned}$$

where x is the system state, z is the system output, w is the zero-mean uncorrelated *plant noise*,Q_{k-1} is its covariance of w_{k-1}, and v is the zero-mean uncorrelated *measurement noise*. Plant noise is also called *process noise*. These models may also have known inputs. *Shaping filters* are models of these types that are used to represent random processes with certain types of spectral properties or temporal correlations.

PROBLEMS

3.1 Let a deck of 52 cards be divided into four piles (labeled North, South, East, West). Find the probability that each pile contains exactly one ace. (There are four aces in all.)

3.2 Show that

$$\binom{n+1}{k+1} = \binom{n}{k+1} + \binom{n}{k}.$$

3.3 How many ways are there to divide a deck of 52 cards into four piles of 13 each?

3.4 If a hand of 13 cards is drawn from a deck of 52, what is the probability that exactly 3 cards are spades? (There are 13 spades in all.)

3.5 If the 52 cards are divided into four piles of 13 each, and if we are told that North has exactly three spades, find the probability that South has exactly three spades.

3.6 A hand of 13 cards is dealt from a well-randomized bridge deck. (The deck contains 13 spades, 13 hearts, 13 diamonds, and 13 clubs.)

(a) What is the probability that the hand contains exactly seven hearts?

(b) During the deal, the face of one of the cards is inadvertently exposed, and it is seen to be a heart. What is now the probability that the hand contains exactly seven hearts?

You may leave the above answers in terms of factorials.

3.7 The random variables $X_1, X_2, \ldots, X_n$ are independent, with mean zero and the same variance σ_X^2. We define the new random variables $Y_1, Y_2, \ldots, Y_n$ by

$$Y_n = \sum_{j=1}^{n} X_j.$$

Find the correlation coefficient r between Y_{n-1} and Y_n.

3.8 The RVs X and Y are independent and uniformly distributed between 0 and 1 (rectangular distribution). Find the probability density function of $Z = |X - Y|$.

3.9 Two RVs X and Y have the density function

$$p_{XY}(x, y) = \begin{cases} C(x - y + 1), & 0 \le y \le x \le 1, \\ 0 & \text{elsewhere,} \end{cases}$$

where the constant $C > 0$ is chosen to normalize the distribution.

(a) Sketch the density function in the x, y plane.
(b) Determine the value of C for normalization.
(c) Obtain two marginal density functions.
(d) Obtain $\mathrm{E}\langle Y|x\rangle$.
(e) Discuss the nature and use of the relation $y = \mathrm{E}\langle Y|x\rangle$.

3.10 The RV $x \in X$ has the probability density function

$$p_X(x) = \begin{cases} 2x, & 0 \le x \le 1, \\ 0 & \text{elsewhere.} \end{cases}$$

Find the following:

(a) The cumulative function $p_X(x)$.
(b) The median.
(c) The mode.
(d) The mean $\mathrm{E}\langle X\rangle$.
(e) The mean-square value $\mathrm{E}\langle X^2\rangle$.
(f) The variance $\sigma^2[X]$.

3.11 An amplitude-modulated signal is specified by

$$y(t) = [1 + mx(t)] \cos(\Omega t + \lambda).$$

Here $x(t)$ is a WSS RP independent of λ, which is an RV uniformly distributed over $[0, 2\pi]$. We are given that

$$\psi_x(\tau) = \frac{1}{\tau^2 + 1}.$$

(a) Verify that $\psi_x(\tau)$ is an autocorrelation.
(b) Let $x(t)$ have the autocorrelation given above. Using the direct method for computing the spectral density, calculate Ψ_y.

3.12 Let $R(T)$ be an arbitrary autocovariance function for a mean-square continuous stochastic process $x(t)$, and let $\Psi(\omega)$ be the PSD for the process $x(t)$. Is it true that

$$\lim_{|\omega| \to \infty} \Psi(\omega) = 0?$$

Justify your answer.

3.13 Find the state-space models for longitudinal, vertical, and lateral turbulence for the following PSD of the Dryden turbulence model:

$$\Psi(\omega) = \sigma^2 \left(\frac{2L}{\pi V}\right)\left(\frac{1}{1 + (L\omega/V)^2}\right)$$

where

$\omega =$ frequency in radians per second
$\sigma =$ root-mean-square turbulence intensity
$L =$ scale length in feet
$V =$ airplane velocity in feet per second (290 ft/s)

(a) For longitudinal turbulence:

$$L = 600\,\text{ft}$$
$$\sigma_u = 0.15 \text{ mean head wind or tail wind (knots)}$$

(b) For vertical turbulence:

$$L = 300 \text{ ft}$$
$$\sigma_w = 1.5 \text{ knots}$$

(c) For lateral turbulence:

$$L = 600 \text{ ft}$$
$$\sigma_v = 0.15 \text{ mean cross-wind (knots)}$$

3.14 Consider the random process

$$x(t) = \cos(\omega_0 t + \theta_1)\cos(\omega_0 t + \theta_2),$$

where θ_1 and θ_2 are independent RVs uniformly distributed between 0 and 2 π.

(a) Show that $x(t)$ is WSS.

(b) Calculate $\psi_x(\tau)$ and $\Psi_x(\omega)$.

(c) Discuss the ergodicity of $x(t)$.

3.15 Let $\psi_x(\tau)$ be the autocovariance of a WSS RP. Is the real part of $\psi_x(\tau)$ necessarily also an autocovariance? If your answer is affirmative, prove it; if negative, give a counterexample.

3.16 Assume $x(t)$ is WSS:

$$y(t) = x(t)\cos(\omega t + \theta),$$

where ω is a constant and θ is a uniformly distributed $[0, 2\pi]$ random phase. Find $\psi_{xy}(\tau)$.

3.17 The RP $x(t)$ has mean zero and autocovariance function

$$\psi_x(\tau) = e^{-|\tau|}.$$

Find the autocovariance function for

$$y(t) = \int_0^t x(u)\,du, \quad t > 0.$$

3.18 Assume $x(t)$ is WSS with PSD

$$\Psi_x(\omega) = \begin{cases} 1, & -a \le \omega \le a, \\ 0 & \text{otherwise.} \end{cases}$$

Sketch the PSD of the process

$$y(t) = x(t)\cos(\Omega t + \theta),$$

where θ is a uniformly distributed random phase and $\Omega > a$.

3.19

(a) Define a WSS RP.
(b) Define a strict-sense stationary RP.
(c) Define a realizable linear system.
(d) Is the following an autocovariance function?

$$\psi(\tau) = \begin{cases} 1 - |\tau|, & |\tau| < 1, \\ 0 & \text{otherwise.} \end{cases}$$

Explain.

3.20 Assume $x(t)$ is a stationary RP with autocovariance function

$$\psi_x(\tau) = \begin{cases} 1 - |\tau|, & -1 \le \tau \le 1, \\ 0 & \text{otherwise.} \end{cases}$$

Find the PSD $\Psi_y(\omega)$ for

$$y(t) = x(t)\cos(\omega_0 t + \lambda)$$

when ω_0 is a constant and λ is an RV uniformly distributed on the interval $[0, 2\pi]$.

3.21 An RP $x(t)$ is defined by

$$x(t) = \cos(t + \theta),$$

where θ is an RV uniformly distributed on the interval $[0, 2\pi]$. Calculate the autocovariance function $\psi_y(t, s)$ for

$$y(t) = \int_0^t x(u)\,du.$$

3.22 Let ψ_1 and ψ_2 be two arbitrary continuous, absolutely integrable autocovariance functions. Are the following necessarily autocovariance functions? Briefly explain your answer.

(a) $\psi_1 \cdot \psi_2$

(b) $\psi_1 + \psi_2$

(c) $\psi_1 - \psi_2$

(d) $\psi_1 * \psi_2$ (the convolution of ψ_1 with ψ_2)

3.23 Give a short reason for each answer:

(a) If $f(t)$ and $g(t)$ are autocovariance functions, $f^2(t) + g(t)$ is (necessarily, perhaps, never) an autocovariance function.

(b) As in (a), $f^2(t) - g(t)$ is (necessarily, perhaps, never) an autocovariance function.

(c) If $x(t)$ is a strictly stationary process, $x^2(t) + 2x(t - 1)$ is (necessarily, perhaps, never) strictly stationary.

(d) The function

$$\omega(\tau) = \begin{cases} \cos\tau, & -\frac{9}{2}\pi \le \tau \le \frac{9}{2}\pi, \\ 0 & \text{otherwise} \end{cases}$$

(is, is not) an autocovariance function.

(e) Let $x(t)$ be strictly stationary and ergodic, and let α be a Gaussian RV with mean 0 and variance 1 and α independent of $x(t)$. Then $y(t) = \alpha x(t)$ is (necessarily, perhaps, never) ergodic.

3.24 Which of the following functions is an autocovariance function of a WSS process? Give a brief reason for each answer.

(a) $e^{-|\tau|}$

(b) $e^{-|\tau|\cos\tau}$

(c) $\Gamma(t) = \begin{cases} 1, & |t| < a, \\ 0 & |t| \ge a \end{cases}$

(d) $e^{-|\tau|\sin\tau}$

(e) $\frac{3}{2}e^{-|\tau|} - e^{-2|\tau|}$

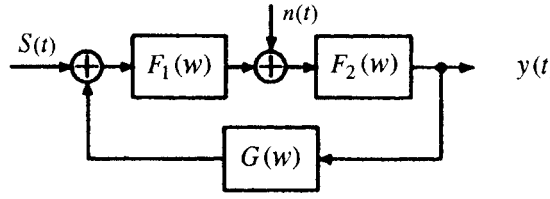
(f) $2e^{-2|\tau|} - e^{-|\tau|}$

3.25 Discuss each of the following:

(a) The distinction between stationarity and WSS.

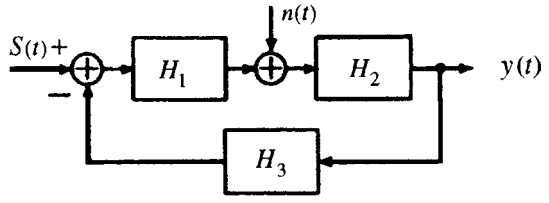
(b) The periodic character of the cross-correlation function of two processes that are themselves periodic, with periods mT and nT, respectively.

3.26 A system transfer function can sometimes be experimentally determined by injecting white noise $n(t)$ and measuring the cross-correlation between the system output and the white noise. Here we consider the following system:



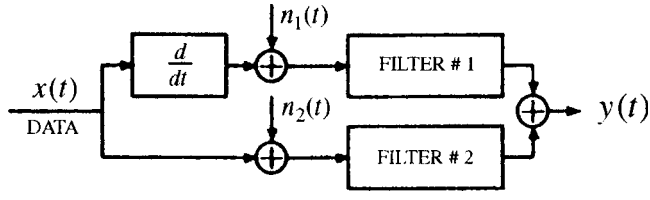
We assume that $\Psi_s(\omega)$ known, $S(t)$ and $n(t)$ are independent, and $\Psi_n(\omega) = 1$. Find $\Psi_{yn}(\omega)$. *Hint:* Write $y(t) = y_S(t) + y_n(t)$, where y_S and y_n are the parts of the output due to S and n, respectively.

3.27 Let $S(t)$ and $n(t)$ be real stationary uncorrelated RPs, each with mean zero.



Here, $H_1(j2\pi\omega)$, $H_2(j2\pi\omega)$, and $H_3(j2\pi\omega)$ are transfer functions of time-invariant linear systems and $S_0(t)$ is the output when $n(t)$ is zero and $n_0(t)$ is the output when $S(t)$ is zero. Find the output signal-to-noise ratio, defined as $\mathrm{E}\,\langle S_0^2(t)\rangle / E\langle n_0^2(t)\rangle$.

3.28 A single random data source is measured by two different transducers, and their outputs are suitably combined into a final measurement $y(t)$. The system is as follows:



Assume that $n_1(t)$ and $n_2(t)$ are uncorrelated RPs, data and noises are uncorrelated, filter 1 has transfer function $Y(s)/s$, and filter 2 has transfer function $1 - Y(s)$. Suppose that it is desired to determine the mean-square error of measurement, where the error is defined by $e(t) = x(t) - y(t)$. Calculate the mean-square value of the error in terms of $Y(s)$ and the PSDs Ψ_x, Ψ_{n_1}, and Ψ_{n_2}.

3.29 Let $x(t)$ be the solution of

$$\dot{x} + x = n(t),$$

where $n(t)$ is white noise with PSD 2π.

(a) Assuming that the above system has been operating since $t = -\infty$, find ψ_x $(t_1\ t_2)$. Investigate whether $x(t)$ is WSS, and if so, express ψ_x accordingly.

(b) Instead of the system in (a), consider

$$\dot{x} + x = \begin{cases} n(t), & t \geq 0, \\ 0, & t < 0, \end{cases}$$

where $x(0) = 0$. Again, compute ψ_x (t_1, t_2).

(c) Let $y(t) = \int_0^t x(\tau)\, d\tau$. Find ψ_{xy} (t_1, t_2) for both of the systems described in (a) and (b).

(d) It is desired to predict $x(t + \alpha)$ from $x(t)$, that is, a future value of the process from its present value. A possible predictor $\hat{x}(t + \alpha)$ is of the form

$$\hat{x}(t + \alpha) = ax(t).$$

Find that a that will give the smallest mean-square prediction error, that is, that minimizes

$$\mathrm{E}\langle |\hat{x}(t + \alpha) - x(t + \alpha)|^2 \rangle,$$

where $x(t)$ is as in part (a).

3.30 Let $x(t)$ be the solution of

$$\dot{x} + x = n(t)$$

with initial condition $x(0) = x_0$. It is assumed that $n(t)$ is white noise with PSD 2π and is turned on at $t = 0$. The initial condition x_0 is an RV independent of $n(t)$ and has zero mean.

(a) If x_0 has variance σ^2, what is $\psi_x(t_1, t_2)$? Derive the result.

(b) Find that value of σ (call it σ_0) for which $\psi_x(t_1, t_2)$ is the same for all $t \geq 0$. Determine whether, with $\sigma = \sigma_0$, $\psi_x(t_1, t_2)$ is a function only of $t_1 - t_2$.

(c) If the white noise had been turned on at $t = -\infty$ and the initial condition has zero mean and variance σ_0^2, as above, is $x(t)$ WSS? Justify your answer by appropriate reasoning and/or computation.

3.31 Let

$$\dot{x}(t) = F(t)\,x(t) + w(t),$$
$$x(a) = x_a, t \geq a,$$

where x_a is a zero-mean RV with covariance matrix P_a and

$$\mathrm{E}\langle w(t)\rangle = 0 \quad \forall t,$$
$$\mathrm{E}\langle w(t)w^{\mathrm{T}}(s)\rangle = Q(t)\delta(t-s) \quad \forall t, s$$
$$\mathrm{E}\langle x(a)w^{\mathrm{T}}(t)\rangle = 0 \quad \forall t.$$

(a) Determine the mean *m(t)* and covariance *P(t, t)* for the process *x(t)*.
(b) Derive a differential equation for *P(t, t)*.

3.32 Find the covariance matrix *P(t)* and its steady-state value $P(\infty)$ for the following continuous systems:

(a) $\dot{x} = \begin{bmatrix} -1 & 0 \\ -1 & 0 \end{bmatrix} x + \begin{bmatrix} 1 \\ 1 \end{bmatrix} w(t), \quad P(0) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

(b) $\dot{x} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} x + \begin{bmatrix} 5 \\ 1 \end{bmatrix} w(t), \quad P(0) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

where $w \in \mathcal{N}(0, 1)$ and white.

3.33 Find the covariance matrix P_k and its steady-state value P_∞ for the following discrete system:

$$x_{k+1} = \begin{bmatrix} 0 & \frac{1}{2} \\ -\frac{1}{2} & 2 \end{bmatrix} x_k + \begin{bmatrix} 1 \\ 1 \end{bmatrix} w_k, \quad P_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},$$

where $w_k \in \mathcal{N}(0, 1)$ and white.

3.34 Find the steady-state covariance for the state-space model given in Example 3.4.

3.35 Show that the continuous-time steady-state algebraic equation

$$0 = FP(\infty) + P(\infty)F^{\mathrm{T}} + GQG^{\mathrm{T}}$$

has no nonnegative solution for the scalar case with $F = Q = G = 1$.

3.36 Show that the discrete-time steady-state algebraic equation

$$P_\infty = \Phi P_\infty \Phi^{\mathrm{T}} + Q$$

has no solution for the scalar case with $\Phi = Q = 1$.

3.37 Find the covariance of x_k as a function of k and its steady-state value for the system

$$x_k = -2x_{k-1} + w_{k-1}$$

where $\mathrm{E}w_{k-1} = 0$ and $\mathrm{E}(w_k w_j) = e^{-|k-j|}$. Assume the initial value of the covariance (P_0) is 1.

3.38 Find the covariance of *x(t)* as a function of t and its steady-state value for the system

$$\dot{x}(t) = -2x(t) + w(t),$$

where $\mathrm{E}w(t) = 0$ and $\mathrm{E}(w(t_1)\, w(t_2)) = e^{-|t_1 - t_2|}$. Assume the initial value of the covariance (P_0) is 1.

3.39 Suppose that *x(t)* has autocovariance function $\psi_x(\tau) = e^{-c|\tau|}$. It is desired to predict $x(t + \alpha)$ on the basis of the past and present of *x(t)*, that is, the predictor may use *x(s)* for all $s \leq t$.

(a) Show that the minimum mean-square error linear prediction is

$$\hat{x}(t + \alpha) = e^{-c\alpha} x(t).$$

(b) Find the mean-square error corresponding to the above. Hint: Use the orthogonality principle.

3.40 Find Q_k in Table 3.4 for

$$F = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

with

(a) Pencil and paper, and

(b) MATLAB.

4

LINEAR OPTIMAL FILTERS AND PREDICTORS

Prediction is difficult—especially of the future.
—Attributed to Niels Henrik David Bohr (1885–1962)

4.1 CHAPTER FOCUS

4.1.1 Estimation Problem

This is the problem of estimating the state of a linear stochastic system by using measurements that are linear functions of the state and measurements.

We suppose that stochastic systems can be represented by the types of plant and measurement models (for continuous and discrete time) shown as Equations 4.1–4.5 in Table 4.1, with dimensions of the vector and matrix quantities as shown in Table 4.2. The symbols $\Delta(k - \ell)$ and $\delta(t - s)$ stand for the *Kronecker delta function* and the *Dirac delta function* (actually, a *generalized* function), respectively.

The measurement and plant noise v_k and w_k are assumed to be zero-mean Gaussian processes, and the initial value x_0 is a Gaussian variate with known mean x_0 and known covariance matrix P_0. Although the noise sequences w_k and v_k are assumed to be uncorrelated, the derivation in Section 4.5 will remove this restriction and modify the estimator equations accordingly.

Kalman Filtering: Theory and Practice using MATLAB®, Third Edition. By Mohinder S. Grewal and Angus P. Andrews

TABLE 4.1 Linear Plant and Measurement Models

Model	Continuous Time	Discrete Time	Equation Number
Plant	$\dot{x}(t) = F(t)x(t) + w(t)$	$x_k = \Phi_{k-1}x_{k-1} + w_{k-1}$	4.1
Measurement	$z(t) = H(t)x(t) + v(t)$	$z_k = H_k x_k + v_k$	4.2
Plant noise	$E\langle w(t)\rangle = 0$	$E\langle w_k\rangle = 0$	
	$E\langle w(t)w^{\mathrm{T}}(s)\rangle = \delta(t-s)Q(t)$	$E\langle w_k w_i^{\mathrm{T}}\rangle = \Delta(k-i)Q_k$	4.3
Observation noise	$E\langle v(t)\rangle = 0$	$E\langle v_k\rangle = 0$	4.4
	$E\langle v(t)v^{\mathrm{T}}(s)\rangle = \delta(t-s)R(t)$	$E\langle v_k v_i^{\mathrm{T}}\rangle = \Delta(k-i)R_k$	4.5

TABLE 4.2 Dimensions of Vectors and Matrices in Linear Models

Symbol	Dimensions
x, w	$n \times 1$
z, v	$\ell \times 1$
R	$\ell \times \ell$
Φ, Q	$n \times n$
H	$\ell \times n$
Δ, δ	Scalar

The objective will be to find an estimate of the n state vector x_k represented by $\hat{x}_k$, a linear function of the measurements $z_i, \ldots, z_k$, that minimizes the weighted mean-squared error

$$E[x_k - \hat{x}_k]^{\mathrm{T}} M[x_k - \hat{x}_k], \tag{4.6}$$

where M is any *symmetric nonnegative-definite weighting matrix*.

4.1.2 Main Points to Be Covered

4.1.2.1 Linear Quadratic Gaussian Estimation Problem We are now prepared to derive the mathematical forms of optimal linear estimators for the states of linear stochastic systems defined in the previous chapters. This is called the *linear quadratic Gaussian* (LQG) estimation problem. The dynamic systems are linear, the performance cost functions are quadratic, and the random processes are Gaussian.

4.1.2.2 Filtering, Prediction, and Smoothing There are three general types of estimators for the LQG problem:

- *Predictors* use observations *strictly prior* to the time that the state of the dynamic system is to be estimated:

$$t_{\mathrm{obs}} < t_{\mathrm{est}}.$$

- *Filters* use observations *up to and including* the time that the state of the dynamic system is to be estimated:

$$t_{\text{obs}} \leq t_{\text{est}}.$$

- *Smoothers* use observations at times t_{obs} *beyond* the time t_{est} that the state of the dynamic system is to be estimated:

$$t_{\text{obs}} > t_{\text{est}}.$$

4.1.2.3 Orthogonality Principle A straightforward and simple approach using the orthogonality principle of Section 3.9.2 is used in the derivation[1] of *estimators*. These estimators will have *minimum variance* and will be *unbiased* and *consistent*.

4.1.2.4 Unbiased Estimators The Kalman filter can be characterized as an algorithm for computing the conditional mean and covariance of the probability distribution of the state of a linear stochastic system with uncorrelated Gaussian process and measurement noise. The conditional mean is the unique *unbiased* estimate. It is propagated in feedback form by a system of linear differential equations or by the corresponding discrete-time equations. The conditional covariance is propagated by a nonlinear differential equation or its discrete-time equivalent. This implementation automatically minimizes the expected risk associated with any quadratic loss function of the estimation error.

4.1.2.5 Performance Properties of Optimal Estimators The statistical performance of the estimator can be predicted *a priori* (i.e., before it is actually used) by solving the nonlinear differential (or difference) equations used in computing the optimal feedback gains of the estimator. These are called *Riccati equations*,[2] and the behavior of their solutions can be shown analytically in the most trivial cases. These equations also provide a means for verifying the proper performance of the actual estimator when it is running.

4.2 KALMAN FILTER

Observational Update Problem for System State Estimator Suppose that a measurement has been made at time t_k and that the information it provides is to be

[1]For more mathematically oriented derivations, consult any of the references, such as Anderson and Moore [8], Bozic [45], Brammer and Siffling [46], Brown [49], Bryson and Ho [52], Bucy and Joseph [56], Catlin [58], Chui and Chen [64], Gelb et al. [95], Jazwinski [116], Kailath [135], Maybeck [183, 184], Mendel [192, 193], Nahi [202], Ruymgaart and Soong [234], and Sorenson [256].

[2]Named in 1763 by Jean le Rond D'Alembert (1717–1783) for Count Jacopo Francesco Riccati (1676–1754), who had studied a second-order scalar differential equation [230], although not the form that we have here [37, 218]. Kalman gives credit to Richard S. Bucy for showing him that the Riccati differential equation is analogous to spectral factorization for defining optimal gains. The Riccati equation also arises naturally in the problem of separation of variables in ordinary differential equations and in the transformation of two-point boundary-value problems into initial-value problems [73].

applied in updating the estimate of the state x of a stochastic system at time t_k. It is assumed that the measurement is linearly related to the state by an equation of the form $z_k = Hx_k + v_k$, where H is the *measurement sensitivity matrix* and v_k is the *measurement noise*.

Estimator in Linear Form The optimal linear estimate is equivalent to the general (nonlinear) optimal estimator if the variates x and z are jointly Gaussian (see Section 3.9.1). Therefore, it suffices to seek an updated estimate $\hat{x}_k(+)$—based on the observation z_k—that is a *linear* function of the *a priori* estimate and the measurement z:

$$\hat{x}_k(+) = K_k^1 \hat{x}_k(-) + \overline{K}_k z_k, \tag{4.7}$$

where $\hat{x}_k(-)$ is the *a priori* estimate of x_k and $\hat{x}_k(+)$ is the *a posteriori* value of the estimate.

Optimization Problem The matrices K_k^1 and $\overline{K}_k$ are as yet unknown. We seek those values of K_k^1 and $\overline{K}_k$ such that the new estimate $\hat{x}_k(+)$ will satisfy the orthogonality principle of Section 3.9.2 This orthogonality condition can be written in the form

$$E\langle [x_k - \hat{x}_k(+)] z_i^{\mathrm{T}} \rangle = 0, \quad i = 1, 2, \ldots, k-1, \tag{4.8}$$

$$E\langle [x_k - \hat{x}_k(+)] z_k^{\mathrm{T}} \rangle = 0. \tag{4.9}$$

If one substitutes the formula for x_k from Equation 4.1 (in Table 4.1) and for $\hat{x}_k(+)$ from Equation 4.7 into Equation 4.8, then one will observe from Equations 4.1 and 4.2 that the data $z_1, \ldots, z_k$ do not involve the noise term w_k. Therefore, because the random sequences w_k and v_k are uncorrelated, it follows that $Ew_k z_i^{\mathrm{T}} = 0$ for $1 \leq i \leq k$. (See Problem 4.5.)

Using this result, one can obtain the following relation:

$$E[\langle \Phi_{k-1} x_{k-1} + w_{k-1} - K_k^1 \hat{x}_k(-) - \overline{K}_k z_k \rangle z_i^T] = 0, \qquad i = 1, \ldots, k-1. \tag{4.10}$$

But because $z_k = H_k x_k + v_k$, Equation 4.10 can be rewritten as

$$E\langle [\Phi_{k-1} x_{k-1} - K_k^1 \hat{x}_k(-) - \overline{K}_k H_k x_k - \overline{K}_k v_k] z_i^{\mathrm{T}} \rangle = 0, \qquad i = 1, \ldots, k-1. \tag{4.11}$$

We also know that Equations 4.8 and 4.9 hold at the previous step, that is,

$$E\langle [x_{k-1} - \hat{x}_{k-1}(+)] z_i^T \rangle = 0, \qquad i = 1, \ldots, k-1,$$

and

$$E\langle v_k z_i^T \rangle = 0, \qquad i = 1, \ldots, k-1.$$

Then Equation 4.11 can be reduced to the form

$$\begin{aligned}
\Phi_{k-1}Ex_{k-1}z_i^{\mathrm{T}} - K_k^1 E\hat{x}_k(-)z_i^{\mathrm{T}} - \overline{K}_k H_k \Phi_{k-1}Ex_{k-1}z_i^{\mathrm{T}} - \overline{K}_k Ev_k z_i^{\mathrm{T}} &= 0,\\
\Phi_{k-1}Ex_{k-1}z_i^{\mathrm{T}} - K_k^1 E\hat{x}_k(-)z_i^{\mathrm{T}} - \overline{K}_k H_k \Phi_{k-1}Ex_{k-1}z_i^{\mathrm{T}} &= 0,\\
E\langle [x_k - \overline{K}_k H_k x_k - K_k^1 x_k] - K_k^1(\hat{x}_k(-) - x_k)\rangle z_i^{\mathrm{T}} &= 0,\\
[1 - K_k^1 - \overline{K}_k H_k]Ex_k z_i^{\mathrm{T}} &= 0. \qquad (4.12)
\end{aligned}$$

Equation 4.12 can be satisfied for any given x_k if

$$K_k^1 = I - \overline{K}_k H_k. \qquad (4.13)$$

Clearly, this choice of K_k^1 causes Equation 4.7 to satisfy a portion of the condition given by Equation 4.8, which was derived in Section 3.9. The choice of $\overline{K}_k$ is such that Equation 4.9 is satisfied.

Let the errors

$$\begin{aligned}
\tilde{x}_k(+) &\triangleq \hat{x}_k(+) - x_k, \qquad (4.14)\\
\tilde{x}_k(-) &\triangleq \hat{x}_k(-) - x_k, \qquad (4.15)\\
\tilde{z}_k &\triangleq \hat{z}_k(-) - z_k,\\
&= H_k\hat{x}_k(-) - z_k. \qquad (4.16)
\end{aligned}$$

Vectors $\tilde{x}_k(+)$ and $x_k(-)$ are the estimation errors after and before updates, respectively.[3]

The variable $\hat{x}_k$ depends linearly on x_k, which depends linearly on z_k. Therefore, from Equation 4.9

$$E[\langle x_k - \hat{x}_k(+)]\tilde{z}_k^{\mathrm{T}}(-)\rangle = 0 \qquad (4.17)$$

and also (by subtracting Equation 4.9 from Equation 4.17)

$$E[x_k - \hat{x}_k(+)]\tilde{z}_k^{\mathrm{T}} = 0. \qquad (4.18)$$

Substitute for x_k, $\hat{x}_k(+)$ and $\tilde{z}_k$ from Equations 4.1, 4.7, and 4.16, respectively. Then

$$E[\Phi_{k-1}x_{k-1} + w_{k-1} - K_k^1\hat{x}_k(-) - \overline{K}_k z_k][H_k\hat{x}_k(-) - z_k]^{\mathrm{T}} = 0.$$

However, by the system structure

$$\begin{aligned}
Ew_k z_k^{\mathrm{T}} = Ew_k\hat{x}_k^{\mathrm{T}}(-) &= 0,\\
E[\Phi_{k-1}x_{k-1} - K_k^1\hat{x}_k(-) - \overline{K}_k z_k][H_k\hat{x}_k(-) - z_k]^{\mathrm{T}} &= 0.
\end{aligned}$$

[3]The symbol $\sim$ is officially called a *tilde* but is often called a *squiggle*.

Substituting for K_k^1, z_k, and $\tilde{x}_k(-)$ and using the fact that $E\tilde{x}_k(-)v_k^{\mathrm{T}} = 0$, this last result can be modified as follows:

$$\begin{aligned}
0 &= E\langle[\Phi_{k-1}x_{k-1} - \hat{x}_k(-) + \overline{K}_k H_k\hat{x}_k(-) - \overline{K}_k H_k x_k - \overline{K}_k v_k] \\
&\quad [H_k\hat{x}_k(-) - H_k x_k - v_k]^{\mathrm{T}}\rangle \\
&= E\langle[(x_k - \hat{x}_k(-)) - \overline{K}_k H_k(x_k - \hat{x}_k(-)) - \overline{K}_k v_k][H_k\tilde{x}_k(-) - v_k]^{\mathrm{T}}\rangle \\
&= E\langle[(-\tilde{x}_k(-) + \overline{K}_k H_k\tilde{x}_k(-) - \overline{K}_k v_k][H_k\tilde{x}_k(-) - v_k]^{\mathrm{T}}\rangle.
\end{aligned}$$

By definition, the *a priori* covariance (the error covariance matrix before the update) is

$$P_k(-) = E\langle\tilde{x}_k(-)\tilde{x}_k^{\mathrm{T}}(-)\rangle.$$

It satisfies the equation

$$[I - \overline{K}_k H_k]P_k(-)H_k^{\mathrm{T}} - \overline{K}_k R_k = 0,$$

and therefore the gain can be expressed as

$$\overline{K}_k = P_k(-)H_k^{\mathrm{T}}[H_k P_k(-)H_k^{\mathrm{T}} + R_k]^{-1}, \tag{4.19}$$

which is the solution we seek for the gain as a function of the *a priori* covariance.

One can derive a similar formula for the *a posteriori* covariance (the error covariance matrix after update), which is defined as

$$P_k(+) = E\langle[\tilde{x}_k(+)\tilde{x}_k^{\mathrm{T}}(+)]\rangle. \tag{4.20}$$

By substituting Equation 4.13 into Equation 4.7, one obtains the equations

$$\begin{aligned}
\hat{x}_k(+) &= (I - \overline{K}_k H_k)\tilde{x}_k(-) + \overline{K}_k z_k, \\
\hat{x}_k(+) &= \tilde{x}_k(-) + \overline{K}_k[z_k - H_k\tilde{x}_k(-)].
\end{aligned} \tag{4.21}$$

Subtract x_k from both sides of the latter equation to obtain the equations

$$\begin{aligned}
\hat{x}_k(+) - x_k &= \hat{x}_k(-) + \overline{K}_k H_k x_k + \overline{K}_k v_k - \overline{K}_k H_k\hat{x}_k(-) - x_k, \\
\tilde{x}_k(+) &= \tilde{x}_k(-) - \overline{K}_k H_k\tilde{x}_k(-) + \overline{K}_k v_k, \\
\tilde{x}_k(+) &= (I - \overline{K}_k H_k)\tilde{x}_k(-) + \overline{K}_k v_k.
\end{aligned} \tag{4.22}$$

By substituting Equation 4.22 into Equation 4.20 and noting that $E\tilde{x}_k(-)v_k^{\mathrm{T}} = 0$, one obtains

$$\begin{aligned}
P_k(+) &= E\{[I - \overline{K}_k H_k]\tilde{x}_k(-)\tilde{x}_k^{\mathrm{T}}(-)[I - \overline{K}_k H_k]^{\mathrm{T}} + \overline{K}_k v_k v_k^{\mathrm{T}}\overline{K}_k^{\mathrm{T}}\} \\
&= (I - \overline{K}_k H_k)P_k(-)(I - \overline{K}_k H_k)^{\mathrm{T}} + \overline{K}_k R_k\overline{K}_k^{\mathrm{T}}.
\end{aligned} \tag{4.23}$$

This last equation is the so-called Joseph form of the covariance update equation derived by P. D. Joseph [56]. By substituting for $\overline{K}$ from Equation 4.19, it can be

put into the following forms:

$$\begin{aligned} P_k(+) &= P_k(-) - \overline{K}_k H_k P_k(-) \\ &\quad - P_k(-) H_k^{\mathrm{T}} \overline{K}_k^{\mathrm{T}} + \overline{K}_k H_k P_k(-) H_k^{\mathrm{T}} \overline{K}_k^{\mathrm{T}} + \overline{K}_k R_k \overline{K}_k^{\mathrm{T}} \\ &= (I - \overline{K}_k H_k) P_k(-) - P_k(-) H_k^{\mathrm{T}} \overline{K}_k^{\mathrm{T}} \\ &\quad + \underbrace{\overline{K}_k (H_k P_k(-) H_k^{\mathrm{T}} + R_k)}_{P_k(-) H_k^{\mathrm{T}}} \overline{K}_k^{\mathrm{T}} \\ &= (I - \overline{K}_k H_k) P_k(-), \end{aligned} \tag{4.24}$$

the last of which is the one most often used in computation. This implements the effect that *conditioning on the measurement* has on the covariance matrix of estimation uncertainty.

Error covariance extrapolation models the effects of time on the covariance matrix of estimation uncertainty, which is reflected in the *a priori* values of the covariance and state estimates,

$$\begin{aligned} P_k(-) &= E[\tilde{x}_k(-) \tilde{x}_k^{\mathrm{T}}(-)], \\ \hat{x}_k(-) &= \Phi_{k-1} \hat{x}_{k-1}(+), \end{aligned} \tag{4.25}$$

respectively. Subtract x_k from both sides of the last equation to obtain the equations

$$\begin{aligned} \hat{x}_k(-) - x_k &= \Phi_{k-1} \hat{x}_{k-1}(+) - x_k, \\ \tilde{x}_k(-) &= \Phi_{k-1} [\hat{x}_{k-1}(+) - x_{k-1}] - w_{k-1} \\ &= \Phi_{k-1} \tilde{x}_{k-1}(+) - w_{k-1} \end{aligned}$$

for the propagation of the estimation error, $\tilde{x}$. Postmultiply it by $\tilde{x}_k^{\mathrm{T}}(-)$ (on both sides of the equation) and take the expected values. Use the fact that $E\tilde{x}_{k-1}\ w_{k-1}^{\mathrm{T}} = 0$ to obtain the results

$$\begin{aligned} P_k(-) &\stackrel{\text{def}}{=} E[\tilde{x}_k(-) \tilde{x}_k^{\mathrm{T}}(-)] \\ &= \Phi_{k-1} E[\tilde{x}_{k-1}(+) \tilde{x}_{k-1}^{\mathrm{T}}(+)] \Phi_{k-1}^{\mathrm{T}} + E[w_{k-1} w_{k-1}^{\mathrm{T}}] \\ &= \Phi_{k-1} P_{k-1}^{(+)} \Phi_{k-1}^{\mathrm{T}} + Q_{k-1}, \end{aligned} \tag{4.26}$$

which gives the *a priori* value of the covariance matrix of estimation uncertainty as a function of the previous *a posteriori* value.

4.2.1 Summary of Equations for the Discrete-Time Kalman Estimator

The equations derived in the previous section are summarized in Table 4.3. In this formulation of the filter equations, G has been combined with the plant covariance

TABLE 4.3 Discrete-Time Kalman Filter Equations

System dynamic model:	$x_k = \Phi_{k-1} + w_{k-1}$ $w_k \sim N(0, Q_k)$
Measurement model:	$z_k = H_k x_k + v_k$ $v_k \sim N(0, R_k)$
Initial conditions:	$E\langle x_0 \rangle = \hat{x}_0$ $E\langle \tilde{x}_0 \tilde{x}_0^T \rangle = P_0$
Independence assumption:	$E\langle w_k v_j^t \rangle = 0$ for all k and j
State estimate extrapolation (Equation 4.25):	$\hat{x}_k(-) = \Phi_{k-1}\hat{x}_{k-1}(+)$
Error covariance extrapolation (Equation 4.26):	$P_k(-) = \Phi_{k-1}P_{k-1}(+)\Phi_{k-1}^{\mathrm{T}} + Q_{k-1}$
State estimate observational update (Equation 4.21):	$\hat{x}_k(+) = \hat{x}_k(-) + \overline{K}_k[z_k - H_k\hat{x}_k(-)]$
Error covariance update (Equation 4.24):	$P_k(+) = [I - \overline{K}_k H_k]P_k(-)$
Kalman gain matrix (Equation 4.19):	$\overline{K}_k = P_k(-)H_k^{\mathrm{T}}[H_k P_k(-)H_k^{\mathrm{T}} + R_k]^{-1}$

by multiplying G_{k-1} and G_{k-1}^{T}, for example,

$$\begin{aligned} Q_{k-1} &= G_{k-1}E(w_{k-1}w_{K-1}^{\mathrm{T}})G_{k-1}^{\mathrm{T}} \\ &= G_{k-1}\overline{Q}_{k-1}G_{k-1}^{\mathrm{T}}. \end{aligned}$$

The relation of the filter to the system is illustrated in the block diagram of Figure 4.1. The basic steps of the computational procedure for the discrete-time Kalman estimator are as follows:

1. Compute $P_k(-)$ using $P_{k-1}(+)$, Φ_{k-1}, and Q_{k-1}.
2. Compute $\overline{K}_k$ using $P_k(-)$ (computed in step 1), H_k, and R_k.
3. Compute $P_k(+)$ using $\overline{K}_k$ (computed in step 2) and $P_k(-)$ (from step 1).
4. Compute successive values of $\hat{x}_k(+)$ recursively using the computed values of $\overline{K}_k$ (from step 3), the given initial estimate $\hat{x}_0$, and the input data z_k.

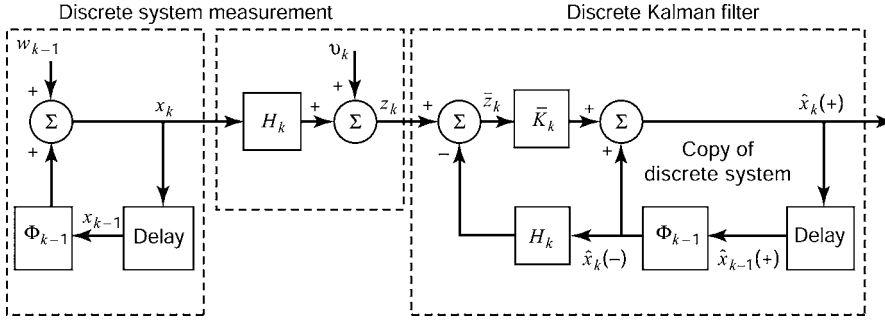


Figure 4.1 Block diagram of a system, a measurement model, and a discrete-time Kalman filter.

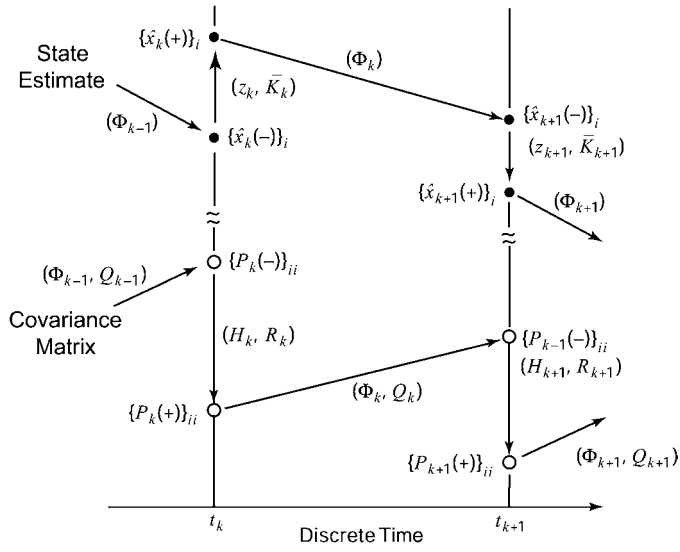


Figure 4.2 Representative sequence of values of filter variables in discrete time.

Step 4 of the Kalman filter implementation [computation of $\hat{x}_k(+)$] can be implemented only for state vector propagation where simulator or real data sets are available. An example of this is given in Section 4.11.

In the design trade-offs, the covariance matrix update (steps 1 and 3) should be checked for symmetry and positive definiteness. Failure to attain either condition is a sign that something is wrong—either a program "bug" or an ill-conditioned problem. In order to overcome ill-conditioning, another equivalent expression for $P_k(+)$ is called the *Joseph form*,[4] as shown in Equation 4.23:

$$P_k(+) = [I - \overline{K}_k H_k]P_k(-)[I - \overline{K}_k H_k]^{\mathrm{T}} + \overline{K}_k R_k \overline{K}_k^{\mathrm{T}}.$$

Note that the right-hand side of this equation is the summation of two symmetric matrices. The first of these is positive definite and the second is nonnegative definite, thereby making $P_k(+)$ a positive definite matrix.

There are many other forms[5] for $\overline{K}_k$ and $P_k(+)$ that might not be as useful for robust computation. It can be shown that state vector update, Kalman gain, and error covariance equations represent an asymptotically stable system, and therefore, the estimate of state $\hat{x}_k$ becomes independent of the initial estimate $\hat{x}_0$, P_0 as k is increased.

Figure 4.2 shows a typical time sequence of values assumed by the ith component of the estimated state vector (plotted with solid circles) and its corresponding variance

[4]After Bucy and Joseph [56].

[5]Some of the alternative forms for computing $\overline{K}_k$ and $P_k(+)$ can be found in Jazwinski [116], Kailath [135], and Sorenson [252].

of estimation uncertainty (plotted with open circles). The arrows show the successive values assumed by the variables, with the annotation (in parentheses) on the arrows indicating which input variables define the indicated transitions. Note that each variable assumes two distinct values at each discrete time: its *a priori* value corresponding to the value *before* the information in the measurement is used and the *a posteriori* value corresponding to the value *after* the information is used.

Example 4.1 (Scalar Problem) Let the system dynamics and observations be given by the following equations:

$$\begin{aligned} x_k &= x_{k-1} + w_{k-1}, z_k = x_k + v_k, \\ E\langle v_k \rangle &= E\langle w_k \rangle = 0, \\ E\langle v_{k_1} v_{k_2} \rangle &= 2\Delta(k_2 - k_1),\ E\langle w_{k_1} w_{k_2} \rangle = \Delta(k_2 - k_1), \\ z_1 &= 2,\ z_2 = 3, \\ E\langle x(0) \rangle &= \hat{x}_0 = 1, \\ E\langle [x(0) - \hat{x}_0][x(0) - \hat{x}_0]^{\mathrm{T}} \rangle &= P_0 = 10. \end{aligned}$$

The objective is to find $\hat{x}_2$ and the steady-state covariance matrix P_∞. One can use the equations in Table 4.3 with

$$\Phi = 1 = H,\ Q = 1,\ R = 2,$$

for which

$$\boxed{P_k(-) = P_{k-1}(+) + 1},$$

$$\overline{K}_k = \frac{P_k(-)}{P_k(-) + 2} = \frac{P_{k-1}(+) + 1}{P_{k-1}(+) + 3},$$

$$P_k(+) = \left[1 - \frac{P_{k-1}(+) + 1}{P_{k-1}(+) + 3}\right](P_{k-1}(+) + 1),$$

$$\boxed{P_k(+) = \frac{2(P_{k-1}(+) + 1)}{P_{k-1}(+) + 3}},$$

$$\boxed{\hat{x}_k(+) = \hat{x}_{k-1}(+) + \overline{K}_k(z_k - \hat{x}_{k-1}(+))}.$$

Let

$$\begin{aligned} P_k(+) = P_{k-1}(+) &= P \text{ (steady-state covariance)}, \\ P &= \frac{2(P+1)}{P+3}, \\ P^2 + P - 2 &= 0, \\ P &= 1, \text{ positive-definite solution.} \end{aligned}$$

For $k = 1$

$$\hat{x}_1(+) = \hat{x}_0 + \frac{P_0 + 1}{P_0 + 3}(2 - \hat{x}_0) = 1 + \frac{11}{13}(2 - 1) = \frac{24}{13}$$

Following is a table for the various values of the Kalman filter:

k	$P_k(-)$	$P_k(+)$	$\overline{K}_k$	$\hat{x}_k(+)$
1	11	$\frac{22}{13}$	$\frac{11}{13}$	$\frac{24}{13}$
2	$\frac{35}{13}$	$\frac{70}{61}$	$\frac{35}{61}$	$\frac{153}{61}$

4.2.2 Treating Vector Measurements with Uncorrelated Errors as Scalars

In many (if not most) applications with vector-valued measurement z, the corresponding matrix R of measurement noise covariance is a diagonal matrix, meaning that the individual components of v_k are uncorrelated. For those applications, it is advantageous to consider the components of z as independent scalar measurements rather than as a vector measurement. The principal advantages are as follows:

1. *Reduced computation time.* The number of arithmetic computations required for processing an ℓ-vector z as ℓ successive scalar measurements is significantly less than the corresponding number of operations for vector measurement processing. It is demonstrated in Chapter 6 that the number of computations for the vector implementation grows as ℓ^3, whereas that of the scalar implementation grows only as ℓ.
2. *Improved numerical accuracy.* Avoiding matrix inversion in the implementation of the covariance equations (by making the expression $HPH^{\mathrm{T}} + R$ a scalar) improves the robustness of the covariance computations against roundoff errors.

The filter implementation in these cases requires ℓ iterations of the observational update equations using the rows of H as measurement "matrices" (with row dimension equal to 1) and the diagonal elements of R as the corresponding (scalar) measurement noise covariance. The updating can be implemented iteratively as the following equations:

$$\overline{K}_k^{[i]} = \frac{1}{H_k^{[i]} P_k^{[i-1]} H_k^{[i]\mathrm{T}} + R_k^{[i]}} P_k^{[i-1]} H_k^{[i]\mathrm{T}},$$

$$P_k^{[i]} = P_k^{[i-1]} - \overline{K}_k^{[i]} H_k^{[i]} P_k^{[i-1]},$$

$$\hat{x}_k^{[i]} = \hat{x}_k^{[i-1]} + \overline{K}_k^{[i]} [\{z_k\}_i - H_k^{[i]} \hat{x}_k^{[i-1]}]$$

for $i = 1, 2, 3, \ldots, \ell$, using the initial values

$$P_k^{[0]} = P_k(-), \quad \hat{x}_k^{[0]} = \hat{x}_k(-);$$

intermediate variables

$$R_k^{[i]} = i\text{th diagonal element of the } \ell \times \ell \text{ diagonal matrix } R_k,$$

$$H_k^{[i]} = i\text{th row of the } \ell \times n \text{ matrix } H_k;$$

and final values

$$P_k^{[\ell]} = P_k(+), \quad \hat{x}_k^{[\ell]} = \hat{x}_k(+).$$

Example 4.2 (Serial Processing of Vector Measurements) Consider the measurement update problem with

$$\hat{x}_{k-} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, P_{k-} = \begin{bmatrix} 4 & 1 \\ 1 & 9 \end{bmatrix}, H_k = \begin{bmatrix} 0 & 2 \\ 3 & 0 \end{bmatrix}, R_k = \begin{bmatrix} 1 & 0 \\ 0 & 4 \end{bmatrix}, z_k = \begin{bmatrix} 3 \\ 4 \end{bmatrix}.$$

Because R is diagonal, the two components of the measurement have independent errors and they can be processed serially, one at a time, as though they were two scalar measurements with mean-squared measurement uncertainties

$$R_1 = 1, R_2 = 4$$

and measurement sensitivity matrices

$$H_1 = [\,0 \quad 2\,], H_2 = [\,3 \quad 0\,].$$

Table 4.4 shows the numerical calculations involved in processing both measurements simultaneously as a vector or as two independent scalar measurements. The implementation equations (left column) include some partial results in square brackets ($[\cdot]$) that are reused to reduce the computational effort required.

The final results are exactly the same by either route, although intermediate results do differ on the two-pass serial processing route. Note, in particular, the following differences:

1. There are intermediate values for the estimated state vector ($\hat{x}_{k+|2}$) and the covariance matrix ($P_{k+|2}$), based on using just the first measurement vector component.
2. The expected value $\hat{z}_2 = H_2\hat{x}_{k+|2}$ on the second pass, based on the intermediate estimate $\hat{x}_{k+|2}$ from using the first scalar measurement, is not the same as the second component of $\hat{z}$ when the measurement is processed as a vector.
3. The covariance matrix P used in computing the Kalman gain $\overline{K}$ on the second pass has the value of $P_{k+|2}$ from the first pass.
4. The two serial Kalman gain vectors bear little resemblence to the two columns of Kalman gain matrix for vector-valued measurements.

TABLE 4.4 Calculations for Example 4.2

Implementation Equations	Vector Measurements	Scalar Measurements	
		1st Meas.	2nd Meas.
$\hat{z} = H\hat{x}_{k-}$	$\begin{bmatrix} 4 \\ 3 \end{bmatrix}$	4	$105/37^*$
$\tilde{z} = z - \hat{z}$	$\begin{bmatrix} -1 \\ 1 \end{bmatrix}$	-1	$43/37$
$[HP]$	$\begin{bmatrix} 2 & 18 \\ 12 & 3 \end{bmatrix}$	$[2 \quad 18]$	$\begin{bmatrix} \frac{432}{37} & \frac{3}{37} \end{bmatrix}^*$
$[[HP]H^T + R]$	$\begin{bmatrix} 37 & 6 \\ 6 & 40 \end{bmatrix}$	37	$1444/37$
$[[HP]H^T + R]^{-1}$	$\begin{bmatrix} \frac{10}{361} & -\frac{3}{722} \\ -\frac{3}{722} & -\frac{37}{1444} \end{bmatrix}$	$1/37$	$37/1444$
$\overline{K} = (HP)^T[HPH^T + R]^{-1}$	$\begin{bmatrix} \frac{2}{361} & \frac{108}{361} \\ \frac{351}{722} & \frac{3}{1444} \end{bmatrix}$	$\begin{bmatrix} \frac{2}{37} \\ \frac{18}{37} \end{bmatrix}$	$\begin{bmatrix} \frac{108}{361} \\ \frac{3}{1444} \end{bmatrix}$
$\hat{x}_{k+} = \hat{x}_{k-} + \overline{K}\,\tilde{z}$	$\begin{bmatrix} \frac{467}{361} \\ \frac{2189}{1444} \end{bmatrix}$	†	$\begin{bmatrix} \frac{467}{361} \\ \frac{2189}{1444} \end{bmatrix}$
$\hat{x}_{k+l2}{}^\dagger$		$\begin{bmatrix} \frac{35}{37} \\ \frac{56}{37} \end{bmatrix}$	
$P_{k+} = P_{k-} - \overline{K}[HP]$	$\begin{bmatrix} \frac{144}{361} & \frac{1}{361} \\ \frac{1}{361} & \frac{351}{1444} \end{bmatrix}$	†	$\begin{bmatrix} \frac{144}{361} & \frac{1}{361} \\ \frac{1}{361} & \frac{351}{1444} \end{bmatrix}$
$P_{k+l2}{}^\dagger$		$\begin{bmatrix} \frac{144}{37} & 1/37 \\ 1/37 & \frac{9}{37} \end{bmatrix}$	

*Note that $\tilde{z}$ on the second pass is based on $\hat{x}_{k+l2}$ from the first pass, not $\hat{x}_{k-}$, and P on the second pass is P_{k+l2} from the first pass.

† Values generated on the first pass are intermediate results.

5. Using the measurement vector components one at a time as independent scalar measurements avoids taking matrix inverses, which significantly reduces the overall amount of computation required.
6. Although writing out the results in this way may make it appear that the two-pass approach takes more computation, the opposite is true. In fact, the advantage of multipass processing of measurement vector components increases with vector length.

4.2.3 Using the Covariance Equations for Design Analysis

It is important to remember that the Kalman gain and error covariance equations are independent of the actual observations. The covariance equations alone are all that is required for characterizing the performance of a proposed sensor system before it is actually built. At the beginning of the design phase of a measurement and estimation system, when neither real nor simulated data are available, just the covariance calculations can be used to obtain preliminary indications of estimator performance. Covariance calculations consist of solving the estimator equations with steps 1–3 in the previous subsection, repeatedly. These covariance calculations will involve the plant noise covariance matrix Q, measurement noise covariance matrix R, state transition matrix Φ, measurement sensitivity matrix H, and initial covariance matrix P_0—all of which must be known for the designs under consideration.

4.3 KALMAN–BUCY FILTER

Analogous to the discrete-time case, the continuous-time random process $x(t)$ and the observation $z(t)$ are given by

$$\dot{x}(t) = F(t)x(t) + G(t)w(t), \tag{4.27}$$

$$z(t) = H(t)x(t) + v(t), \tag{4.28}$$

$$Ew(t) = Ev(t) = 0,$$

$$Ew(t_1)w^{\mathrm{T}}(t_2) = Q(t)\delta(t_2 - t_1), \tag{4.29}$$

$$Ev(t_1)v^{\mathrm{T}}(t_2) = R(t)\delta(t_2 - t_1), \tag{4.30}$$

$$Ew(t)v^{\mathrm{T}}(\eta) = 0, \tag{4.31}$$

where $F(t)$, $G(t)$, $H(t)$, $Q(t)$, and $R(t)$ are $n \times n$, $n \times n$, $l \times n$, $n \times n$, and $l \times l$ matrices, respectively. The term $\delta(t_2 - t_1)$ is the Dirac delta. The covariance matrices Q and R are positive definite.

It is desired to find the estimate of n state vector $x(t)$ represented by $\hat{x}(t)$ which is a linear function of the measurements $z(t)$, $0 \leq t \leq T$, which minimizes the scalar equation

$$E[x(t) - \hat{x}(t)]^{\mathrm{T}} M[x(t) - \hat{x}(t)], \tag{4.32}$$

where M is a symmetric positive-definite matrix.

The initial estimate and covariance matrix are $\hat{x}_0$ and P_0.

This section provides a formal derivation of the continuous-time Kalman–Bucy estimator. A rigorous derivation can be achieved by using the orthogonality principle as in the discrete-time case. In view of the main objective (to obtain efficient and practical estimators), less emphasis is placed on continuous-time estimators.

Let Δt be the time interval $[t_k - t_{k-1}]$. As shown in Chapters 2 and 3, the following relationships are obtained:

$$\Phi(t_k,t_{k-1}) = \Phi_k = I + F(t_{k-1})\Delta t + 0(\Delta t^2),$$

where $0(\Delta t^2)$ consists of terms with powers of Δt greater than or equal to 2. For measurement noise

$$R_k = \frac{R(t_k)}{\Delta t},$$

and for process noise

$$Q_k = G(t_k)Q(t_k)G^{\mathrm{T}}(t_k)\Delta t.$$

Equations 4.24 and 4.26 can be combined. By substituting the above relations, one can get the result

$$\begin{aligned} P_k(-) &= [I + F(t)\Delta t][I - \overline{K}_{k-1}H_{k-1}]P_{k-1}(-) \\ &\quad \times [I + F(t)\Delta t]^{\mathrm{T}} + G(t)Q(t)G^{\mathrm{T}}(t)\Delta t, \end{aligned} \tag{4.33}$$

$$\begin{aligned} \frac{P_k(-) - P_{k-1}(-)}{\Delta t} &= F(t)P_{k-1}(-) + P_{k-1}(-)F^{\mathrm{T}}(t) \\ &\quad + G(t)Q(t)G^{\mathrm{T}}(t) - \frac{\overline{K}_{k-1}H_{k-1}P_{k-1}(-)}{\Delta t} \\ &\quad - F(t)\overline{K}_{k-1}H_{k-1}P_{k-1}(-)F^{\mathrm{T}}(t)\Delta t \\ &\quad + \text{higher order terms}. \end{aligned} \tag{4.34}$$

The Kalman–Bucy gain of Equation 4.19 becomes, in the limit,

$$\begin{aligned} \lim_{\Delta t\to 0}\left[\frac{\overline{K}_{k-1}}{\Delta t}\right] &= \lim_{\Delta t\to 0}\{P_{k-1}(-)H_{k-1}^{\mathrm{T}}[H_{k-1}P_{k-1}(-)H_{k-1}^{\mathrm{T}}\Delta t + R(t)]^{-1}\} \\ &= PH^{\mathrm{T}}R^{-1} = \overline{K}(t). \end{aligned} \tag{4.35}$$

Substituting Equation 4.35 in 4.34 and taking the limit as $\Delta t \to 0$, one obtains the desired result

$$\begin{aligned} \dot{P}(t) &= F(t)P(t) + P(t)F^{\mathrm{T}}(t) + G(t)Q(t)G^{\mathrm{T}}(t) \\ &\quad - P(t)H^{\mathrm{T}}(t)R^{-1}(t)H(t)P(t) \end{aligned} \tag{4.36}$$

with $P(t_0)$ as the initial condition. This is called the *matrix Riccati differential equation*. Methods for solving it will be discussed in Section 4.8. The differential equation can be rewritten by using the identity

$$P(t)H^{\mathrm{T}}(t)R^{-1}(t)R(t)R^{-1}(t)H(t)P(t) = \overline{K}(t)R(t)\overline{K}^{\mathrm{T}}(t)$$

to transform Equation 4.36 into the form

$$\dot{P}(t) = F(t)P(t) + P(t)F^{\mathrm{T}}(t) + G(t)Q(t)G^{\mathrm{T}}(t) - \overline{K}(t)R(t)\overline{K}^{\mathrm{T}}(t). \tag{4.37}$$

In similar fashion, the state vector update equation can be derived from Equations 4.21 and 4.25 by taking the limit as $\Delta t \to 0$ to obtain the differential equation for the estimate:

$$\hat{x}_k(+) = \Phi_{k-1}\hat{x}_{k-1}(+) + \bar{K}[z_k - H_k\Phi_{k-1}x_{k-1}(+)] \tag{4.38}$$

$$\approx [I + F\,\Delta t]\hat{x}_{k-1}(+) + \bar{K}_k[z_k - H_k(I + F\,\Delta t)\hat{x}_{k-1}(+)] \tag{4.39}$$

$$\dot{\hat{x}}(t_k) = \lim_{\Delta t \to 0} \frac{x_k(+) - x_{k-1}(+)}{\Delta t} \tag{4.40}$$

$$= \lim_{\Delta t \to 0} \left[F\hat{x}_{k-1}(+) \frac{\bar{K}_k}{\Delta t}(z_k - H_k\hat{x}_{k-1}(+) - H_kF_k\,\Delta t\,\hat{x}_{k-1}(+)) \right] \tag{4.41}$$

$$= F(t)\hat{x}(t) + \overline{K}(t)[z(t) - H(t)\hat{x}(t)] \tag{4.42}$$

with initial condition $\hat{x}(0)$. Equations 4.35, 4.37, and 4.42 define the continuous-time Kalman–Bucy estimator, which is also called the *Kalman–Bucy filter* [159, 143, 145, 146].

4.4 OPTIMAL LINEAR PREDICTORS

4.4.1 Prediction as Filtering

Prediction is equivalent to filtering when measurement data are unavailable or unreliable. In such cases, the Kalman gain matrix $\overline{K}_k$ is forced to be zero. Hence, Equations 4.21, 4.25, and 4.42 become

$$\hat{x}_k(+) = \Phi_{k-1}\hat{x}_{k-1}(+) \tag{4.43}$$

and

$$\dot{\hat{x}}(t) = F(t)\hat{x}(t). \tag{4.44}$$

Previous values of the estimates will become the initial conditions for the above equations.

4.4.2 Accommodating Missing Data

It sometimes happens in practice that measurements that had been scheduled to occur over some time interval ($t_{k_1} < t \leq t_{k_2}$) are, in fact, unavailable or unreliable. The estimation accuracy will suffer from the missing information, but thefilter can continue to operate without modification. One can continue using the prediction algorithm given in Section 4.4 to continually estimate x_k for $k > k_1$ using the last available estimate $\hat{x}_{k_1}$ until the measurements again become useful (after $k = k_2$).

It is unnecessary to perform the observational update, because there is no information on which to base the conditioning. In practice, the filter is often run with the measurement sensitivity matrix $H = 0$ so that, in effect, the only update performed is the temporal update.

4.5 CORRELATED NOISE SOURCES

4.5.1 Correlation between Plant and Measurement Noise

We want to consider the extensions of the results given in Sections 4.2 and 4.3, allowing correlation between the two noise processes (assumed jointly Gaussian). Let the correlation be given by

$$Ew_{k_1} v_{k_2}^{\mathrm{T}} = C_{k_1} \Delta(k_2 - k_1) \quad \text{for the discrete-time case,}$$
$$Ew(t_1)v^{\mathrm{T}}(t_2) = C(t_1)\delta(t_2 - t_1) \quad \text{for the continuous-time case.}$$

For this extension, the discrete-time estimators have the same initial conditions and state estimate extrapolation and error covariance extrapolation equations. However, the measurement update equations in Table 4.3 will be modified to

$$\begin{aligned} \overline{K}_k &= [P_k(-)H_k^{\mathrm{T}} + C_k][H_k P_k(-)H_k^{\mathrm{T}} + R_k + H_k C_k + C_k^{\mathrm{T}} H^{\mathrm{T}}]^{-1}, \\ P_k(+) &= P_k(-) - \overline{K}_k[H_k P_k(-) + C_k^{\mathrm{T}}], \\ \hat{x}_k(+) &= \hat{x}_k(-) + \overline{K}_k[z_k - H_k \hat{x}_k(-)]. \end{aligned} \tag{4.45}$$

Similarly, the continuous-time estimator algorithms can be extended to include the correlation. Equation 4.35 is changed as follows [53, 250]:

$$\overline{K}(t) = [P(t)H^{\mathrm{T}}(t) + C(t)]R^{-1}(t). \tag{4.46}$$

4.5.2 Time-Correlated Measurements

Correlated measurement noise v_k can be modeled by a shaping filter driven by white Gaussian noise (see Section 3.6). Let the measurement model be given by

$$z_k = H_k x_k + v_k,$$

where

$$v_k = A_{k-1} v_{k-1} + \eta_{k-1} \tag{4.47}$$

and η_k is zero-mean white Gaussian noise with covarience $Q\eta$.

Equation 4.1 is augmented by Equation 4.47, and the new state vector $X_k = [x_k^{\mathrm{T}} \; v_k^{\mathrm{T}}]^{\mathrm{T}}$ satisfies the difference equation:

$$X_k = \begin{bmatrix} x_k \\ -- \\ v_k \end{bmatrix} = \begin{bmatrix} \Phi_{k-1} & 0 \\ - - - & - - - - \\ 0 & A_{k-1} \end{bmatrix} \begin{bmatrix} x_{k-1} \\ -- \\ v_{k-1} \end{bmatrix} + \begin{bmatrix} w_{k-1} \\ - - - \\ \eta_{k-1} \end{bmatrix},$$

$$z_k = [H_k \vdots I] X_k.$$

White noise is so pervasive in real-world electromechanical and electronic systems that there is almost surely a white noise component present in the outputs of any sensor. In discrete-time applications, for example, *quantization noise* from sample digitization is usually modeled as Gaussian white noise with $R_k = \Delta^2/12$, where Δ is the quantization step size. For analog sensors, R_k can usually be inferred from the "white noise floor" in the sensor output PSD beyond the bandwidth of the correlated noise.

4.6 RELATIONSHIPS BETWEEN KALMAN–BUCY AND WIENER FILTERS

The Wiener filter is defined for stationary systems in continuous time, and the Kalman filter is defined for either stationary or nonstationary systems in either discrete time or continuous time, but with finite state dimension. To demonstrate the connections on problems satisfying both sets of constraints, take the continuous-time Kalman–Bucy estimator equations of Section 4.3, letting F, G, and H be constants, the noises be stationary (Q and R constant), and the filter reach steady state (P constant). That is, as $t \to \infty$, $\dot{P}(t) \to 0$. The Riccati differential equation from Section 4.3 becomes the algebraic Riccati equation

$$0 = FP(\infty) + P(\infty)F^{\mathrm{T}} + GQG^{\mathrm{T}} - P(\infty)H^{\mathrm{T}}R^{-1}HP(\infty)$$

for continuous-time systems. The positive-definite solution of this algebraic equation is the steady-state value of the covariance matrix, $[P(\infty)]$. The Kalman–Bucy filter equation in steady state is then

$$\dot{\hat{x}}(t) = F\hat{x} + \overline{K}(\infty)[z(t) - H\hat{x}(t)].$$

Take the Laplace transform of both sides of this equation, assuming that the initial conditions are equal to zero, to obtain the following transfer function:

$$[sI - F + \overline{K}H]\hat{x}(s) = \overline{K}z(s),$$

where the Laplace transforms $\mathcal{L}\hat{x}(t) = \hat{x}(s)$ and $\mathcal{L}z(t) = z(s)$. This has the solution

$$\hat{x}(s) = [sI - F + \overline{K}H]^{-1}\overline{K}z(s),$$

where the steady-state gain

$$\overline{K} = P(\infty)H^{\mathrm{T}}R^{-1}.$$

This transfer function represents the steady-state Kalman–Bucy filter, which is identical to the Wiener filter [183].

4.7 QUADRATIC LOSS FUNCTIONS

The Kalman filter minimizes *any* quadratic loss function of estimation error. Just the fact that it is *unbiased* is sufficient to prove this property, but saying that the estimate is unbiased is equivalent to saying that $\hat{x} = E(x)$. That is, the estimated value is the *mean* of the probability distribution of the state.

4.7.1 Quadratic Loss Functions of Estimation Error

A *loss function* or *penalty function*[6] is a real-valued function of the outcome of a random event. A loss function reflects the *value* of the outcome. Value concepts can be somewhat subjective. In gambling, for example, your perceived loss function for the outcome of a bet may depend upon your personality and current state of winnings, as well as on how much you have riding on the bet.

4.7.1.1 Loss Functions of Estimates In estimation theory, the perceived loss is generally a function of *estimation error* (the difference between an estimated function of the outcome and its actual value), and it is generally a monotonically increasing function of the absolute value of the estimation error. In other words, bigger errors are valued less than smaller errors.

4.7.1.2 Quadratic Loss Functions If x is a real n-vector (variate) associated with the outcome of an event and $\hat{x}$ is an estimate of x, then a quadratic loss function for the estimation error $\hat{x} - x$ has the form

$$L(\hat{x} - x) = (\hat{x} - x)^{\mathrm{T}}M(\hat{x} - x), \tag{4.48}$$

[6]These are concepts from decision theory, which includes estimation theory. The theory might have been based just as well on more optimistic concepts, such as *gain functions*, *benefit functions*, or *reward functions*, but the nomenclature seems to have been developed by pessimists. This focus on the negative aspects of the problem is unfortunate, and you should not allow it to dampen your spirit.

where M is a *symmetric positive-definite matrix*. One may as well assume that M is symmetric, because the skew-symmetric part of M does not influence the quadratic loss function. The reason for assuming positive definiteness is to ensure that the loss is zero only if the error is zero, and the loss is a monotonically increasing function of the absolute estimation error.

4.7.2 Expected Value of a Quadratic Loss Function

4.7.2.1 Loss and Risk The expected value of loss is sometimes called *risk*. It will be shown that the expected value of a quadratic loss function of the estimation error $\hat{x}-x$ is a quadratic function of $\hat{x} - E\langle x\rangle$, where $E\langle\hat{x}\rangle = E\langle x\rangle$. This demonstration will depend upon the following identities:

$$\hat{x} - x = (\hat{x} - E\langle x\rangle) - (x - E\langle x\rangle), \tag{4.49}$$

$$E_x\langle x - E\langle x\rangle\rangle = 0, \tag{4.50}$$

$$E_x\langle(x - E\langle x\rangle)^{\mathrm{T}}M(x - E\langle x\rangle)\rangle = E_x\langle\mathrm{trace}[(x - E\langle x\rangle)^{\mathrm{T}}M(x - E\langle x\rangle)]\rangle \tag{4.51}$$

$$= E_x\langle\mathrm{trace}[M(x - E\langle x\rangle)(x - E\langle x\rangle)^{\mathrm{T}}]\rangle \tag{4.52}$$

$$= \mathrm{trace}[ME_x\langle(x - E\langle x\rangle)(x - E\langle x\rangle)^{\mathrm{T}}] \tag{4.53}$$

$$= \mathrm{trace}[MP], \tag{4.54}$$

$$P \overset{\mathrm{def}}{=} E_x\langle(x - E\langle x\rangle)(x - E\langle x\rangle)^{\mathrm{T}}\rangle. \tag{4.55}$$

4.7.2.2 Risk of a Quadratic Loss Function In the case of the quadratic loss function defined above, the expected loss (risk) will be

$$\mathcal{R}(\hat{x}) = E_x\langle L(\hat{x} - x)\rangle \tag{4.56}$$

$$= E_x\langle(\hat{x} - x)^{\mathrm{T}}M(\hat{x} - x)\rangle \tag{4.57}$$

$$= E_x\langle[(\hat{x} - E\langle x\rangle) - (x - E\langle x\rangle)]^{\mathrm{T}}M[(\hat{x} - E\langle x\rangle) - (x - E\langle x\rangle)]\rangle \tag{4.58}$$

$$\begin{aligned} &= E_x\langle(\hat{x} - E\langle x\rangle)^{\mathrm{T}}M(\hat{x} - E\langle x\rangle) + (x - E\langle x\rangle)^{\mathrm{T}}M(x - E\langle x\rangle)\rangle \\ &\quad - E_x\langle(\hat{x} - E\langle x\rangle)^{\mathrm{T}}M(x - E\langle x\rangle) - (x - E\langle x\rangle)^{\mathrm{T}}M(\hat{x} - E\langle x\rangle)\rangle \end{aligned} \tag{4.59}$$

$$\begin{aligned} &= (\hat{x} - E\langle x\rangle)^{\mathrm{T}}M(\hat{x} - E\langle x\rangle) + E_x\langle(x - E\langle x\rangle)^{\mathrm{T}}M(x - E\langle x\rangle)\rangle \\ &\quad - (\hat{x} - E\langle x\rangle)^{\mathrm{T}}ME_x\langle(x - E\langle x\rangle)\rangle - E_x\langle(x - E\langle x\rangle)\rangle^{\mathrm{T}}M(\hat{x} - E\langle x\rangle) \end{aligned} \tag{4.60}$$

$$= (\hat{x} - E\langle x\rangle)^{\mathrm{T}}M(\hat{x} - E\langle x\rangle) + \mathrm{trace}[MP], \tag{4.61}$$

which is a quadratic function of $\hat{x} - E\langle x\rangle$ with the added nonnegative[7] constant trace$[MP]$.

[7]Recall that M and P are symmetric and nonnegative definite, and that the matrix trace of any product of symmetric nonnegative definite matrices is nonnegative.

4.7.3 Unbiased Estimates and Quadratic Loss

The *estimate* $\hat{x} = E\langle x \rangle$ minimizes the expected value of any *positive-definite quadratic loss function*. From the above derivation,

$$\mathcal{R}(\hat{x}) \geq \text{trace}[MP] \tag{4.62}$$

and

$$\mathcal{R}(\hat{x}) = \text{trace}[MP], \tag{4.63}$$

only if

$$\hat{x} = E\langle x \rangle, \tag{4.64}$$

where it has been assumed only that the mean $E\langle x \rangle$ and covariance $E_x\langle (x\text{-}E\langle x \rangle)(x - E\langle x \rangle)^{\mathrm{T}} \rangle$ are defined for the probability distribution of x. This demonstrates the utility of quadratic loss functions in estimation theory: They always lead to the mean as the estimate with minimum expected loss (risk).

4.7.3.1 Unbiased Estimates An estimate $\hat{x}$ is called *unbiased* if the expected estimation error $E_x\langle \hat{x} - x \rangle = 0$. What has just been shown is that an unbiased estimate minimizes the expected value of any quadratic loss function of estimation error.

4.8 MATRIX RICCATI DIFFERENTIAL EQUATION

The need to solve the Riccati equation is perhaps the greatest single cause of anxiety and agony on the part of people faced with implementing a Kalman filter. This section presents a brief discussion of solution methods for the Riccati *differential* equation for the Kalman–Bucy filter. An analogous treatment of the discrete-time problem for the Kalman filter is presented in the next section. A more thorough treatment of the Riccati equation can be found in the book by Bittanti et al. [37].

4.8.1 Transformation to a Linear Equation

The Riccati differential equation was first studied in the eighteenth century as a nonlinear scalar differential equation, and a method was derived for transforming it to a linear matrix differential equation. That same method works when the dependent variable of the original Riccati differential equation is a matrix. That solution method is derived here for the matrix Riccati differential equation of the Kalman–Bucy filter. An analogous solution method for the discrete-time matrix Riccati equation of the Kalman filter is derived in the next section.

4.8.1.1 Matrix Fractions A matrix product of the sort AB^{-1} is called a *matrix fraction*, and a representation of a matrix M in the form

$$M = AB^{-1}$$

will be called a *fraction decomposition* of M. The matrix A is the *numerator* of the fraction, and the matrix B is its *denominator*. It is necessary that the matrix denominator be nonsingular.

4.8.1.2 Linearization by Fraction Decomposition The Riccati differential equation is nonlinear. However, a fraction decomposition of the covariance matrix results in a linear differential equation for the numerator and denominator matrices. The numerator and denominator matrices will be functions of time, such that the product $A(t)B^{-1}(t)$ satisfies the matrix Riccati differential equation and its boundary conditions.

4.8.1.3 Derivation By taking the derivative of the matrix fraction $A(t)B^{-1}(t)$ with respect to t and using the fact[8] that

$$\frac{d}{dt}B^{-1}(t) = -B^{-1}(t)\dot{B}(t)B^{-1}(t),$$

one can arrive at the following decomposition of the matrix Riccati differential equation, where GQG^{T} has been reduced to an equivalent Q:

$$\dot{A}(t)B^{-1}(t) - A(t)B^{-1}(t)\dot{B}(t)B^{-1}(t) = \frac{d}{dt}\{A(t)B^{-1}(t)\} \tag{4.65}$$

$$= \frac{d}{dt}P(t) \tag{4.66}$$

$$= F(t)P(t) + P(t)F^{\mathrm{T}}(t) - P(t)H^{\mathrm{T}}(t)R^{-1}(t)H(t)P(t) + Q(t) \tag{4.67}$$

$$= F(t)A(t)B^{-1}(t) + A(t)B^{-1}(t)F^{\mathrm{T}}(t) - A(t)B^{-1}(t)H^{\mathrm{T}}(t)R^{-1}(t)H(t)A(t)B^{-1}(t) + Q(t), \tag{4.68}$$

$$\dot{A}(t) - A(t)B^{-1}(t)\dot{B}(t) = F(t)A(t) + A(t)B^{-1}(t)F^{\mathrm{T}}(t)B(t) - A(t)B^{-1}(t)H^{\mathrm{T}}(t)R^{-1}(t)H(t)A(t) + Q(t)B(t), \tag{4.69}$$

$$\dot{A}(t) - A(t)B^{-1}(t)\{\dot{B}(t)\} = F(t)A(t) + Q(t)B(t) - A(t)B^{-1}(t) \times \{H^{\mathrm{T}}(t)R^{-1}(t)H(t)A(t) - F^{\mathrm{T}}(t)B(t)\}, \tag{4.70}$$

$$\dot{A}(t) = F(t)A(t) + Q(t)B(t), \tag{4.71}$$

$$\dot{B}(t) = H^{\mathrm{T}}(t)R^{-1}(t)H(t)A(t) - F^{\mathrm{T}}(t)B(t), \tag{4.72}$$

$$\frac{d}{dt}\begin{bmatrix} A(t) \\ B(t) \end{bmatrix} = \begin{bmatrix} F(t) & Q(t) \\ H^{\mathrm{T}}(t)R^{-1}(t)H(t) & -F^{\mathrm{T}}(t) \end{bmatrix}\begin{bmatrix} A(t) \\ B(t) \end{bmatrix}. \tag{4.73}$$

[8]This formula is derived in Appendix B, Equation B.10.

The last equation is a linear first-order matrix differential equation. The dependent variable is a $2n \times n$ matrix, where n is the dimension of the underlying state variable.

4.8.1.4 Hamiltonian Matrix This is the name[9] given the matrix

$$\Psi(t) = \begin{bmatrix} F(t) & Q(t) \\ H^{\mathrm{T}}(t)R^{-1}(t)H(t) & -F^{\mathrm{T}}(t) \end{bmatrix} \tag{4.74}$$

of the matrix Riccati differential equation.

4.8.1.5 Boundary Constraints The initial values of $A(t)$ and $B(t)$ must also be constrained by the initial value of $P(t)$. This is easily satisfied by taking $A(t_0) = P(t_0)$ and $B(t_0) = I$, the identity matrix.

4.8.2 Time-Invariant Problem

In the time-invariant case, the Hamiltonian matrix Ψ is also time-invariant. As a consequence, the solution for the numerator A and denominator B of the matrix fraction can be represented in matrix form as the product

$$\begin{bmatrix} A(t) \\ B(t) \end{bmatrix} = e^{\Psi t} \begin{bmatrix} P(0) \\ I \end{bmatrix},$$

where $e^{\Psi t}$ is a $2n \times 2n$ matrix.

4.8.3 Scalar Time-Invariant Problem

For this problem, the numerator A and denominator B of the *matrix fraction* AB^{-1} will be scalars, but Ψ will be a 2×2 matrix. We will here show how its exponential can be obtained in closed form. This will illustrate an application of the linearization procedure, and the results will serve to illuminate properties of the solutions—such as their dependence on initial conditions and on the scalar parameters F, H, R, and Q.

4.8.3.1 Linearizing the Differential Equation The scalar time-invariant Riccati differential equation and its linearized equivalent are

$$\dot{P}(t) = FP(t) + P(t)F - P(t)HR^{-1}HP(t) + Q,$$

$$\begin{bmatrix} \dot{A}(t) \\ \dot{B}(t) \end{bmatrix} = \begin{bmatrix} F & Q \\ HR^{-1}H & -F \end{bmatrix} \begin{bmatrix} A(t) \\ B(t) \end{bmatrix},$$

[9]After the Irish mathematician and physicist William Rowan Hamilton (1805–1865).

respectively, where the symbols F, H, R, and Q represent scalar parameters (constants) of the application, t is a free (independent) variable, and the dependent variable P is constrained as a function of t by the differential equation. One can solve this equation for P as a function of the free variable t *and* as a function of the parameters F, H, R, and Q.

4.8.3.2 Fundamental Solution of the Linear Time-Invariant Differential Equation The linear time-invariant differential equation has the general solution

$$\begin{bmatrix} A(t) \\ B(t) \end{bmatrix} = e^{\Psi t} \begin{bmatrix} P(0) \\ 1 \end{bmatrix}, \qquad \Psi = \begin{bmatrix} F & Q \\ \dfrac{H^2}{R} & -F \end{bmatrix}. \tag{4.75}$$

This matrix exponential will now be evaluated by using the characteristic vectors of Ψ, which are arranged as the column vectors of the matrix

$$M = \begin{bmatrix} \dfrac{-Q}{F+\phi} & \dfrac{-Q}{F-\phi} \\ 1 & 1 \end{bmatrix}, \quad \phi = \sqrt{F^2 + \frac{H^2 Q}{R}},$$

with inverse

$$M^{-1} = \begin{bmatrix} \dfrac{-H^2}{2\phi R} & \dfrac{\phi + F}{2\phi} \\ \dfrac{H^2}{2\phi R} & \dfrac{\phi - F}{2\phi} \end{bmatrix},$$

by which it can be diagonalized as

$$M^{-1}\Psi M = \begin{bmatrix} \lambda_2 & 0 \\ 0 & \lambda_1 \end{bmatrix}, \qquad \lambda_2 = -\phi, \quad \lambda_1 = \phi,$$

with the characteristic values of Ψ along its diagonal. The exponential of the diagonalized matrix, multiplied by t, will be

$$e^{M^{-1}\Psi M t} = \begin{bmatrix} e^{\lambda_2 t} & 0 \\ 0 & e^{\lambda_1 t} \end{bmatrix}.$$

Using this, one can write the fundamental solution of the linear homogeneous time-invariant equation as

$$
\begin{aligned}
e^{\Psi t} &= \sum_{k=0}^{\infty} \frac{1}{k!} t^k \Psi^k \\
&= M \left(\sum_{k=0}^{\infty} \frac{1}{k!} [M^{-1} \Psi M]^k \right) M^{-1} \\
&= M e^{M^{-1} \Psi M t} M^{-1} \\
&= M \begin{bmatrix} e^{\lambda_2 t} & 0 \\ 0 & e^{\lambda_1 t} \end{bmatrix} M^{-1} \\
&= \frac{1}{2e^{\phi t}\phi} \begin{bmatrix} \phi(\psi(t)+1) + F(\psi(t)-1) & -Q(1-\psi(t)) \\ \dfrac{H^2(\psi(t)-1)}{R} & F(1-\psi(t)) + \phi(1+\psi(t)) \end{bmatrix}, \\
\psi(t) &= e^{2\phi t}
\end{aligned}
$$

and the solution of the linearized system as

$$
\begin{aligned}
\begin{bmatrix} A(t) \\ B(t) \end{bmatrix} &= e^{\Psi t} \begin{bmatrix} P(0) \\ 1 \end{bmatrix} \\
&= \frac{1}{2e^{\phi t}\phi} \begin{bmatrix} P(0)[\phi(\psi(t)+1) + F(\psi(t)-1)] + Q(\psi(t)-1) \\ \dfrac{P(0)H^2(\psi(t)-1)}{R} + \phi(\psi(t)+1) - F(\psi(t)-1) \end{bmatrix}.
\end{aligned}
$$

4.8.3.3 General Solution of the Scalar Time-Invariant Riccati Equation The general solution formula may now be composed from the previous results as

$$
\begin{aligned}
P(t) &= A(t)/B(t) \\
&= \frac{\mathcal{N}_P(t)}{\mathcal{D}_P(t)}, \qquad (4.76) \\
\mathcal{N}_P(t) &= R[P(0)(\phi + F) + Q] + R[P(0)(\phi - F) - Q]e^{-2\phi t} \\
&= R\left[P(0)\left(\sqrt{F^2 + \frac{H^2 Q}{R}} + F \right) + Q \right] \\
&\quad + R\left[P(0)\left(\sqrt{F^2 + \frac{H^2 Q}{R}} - F \right) - Q \right] e^{-2\phi t}, \qquad (4.77) \\
\mathcal{D}_P(t) &= [H^2 P(0) + R(\phi - F)] - [H^2 P(0) - R(F + \phi)]e^{-2\phi t} \\
&= \left[H^2 P(0) + R\left(\sqrt{F^2 + \frac{H^2 Q}{R}} - F \right) \right] \\
&\quad - \left[H^2 P(0) - R\left(\sqrt{F^2 + \frac{H^2 Q}{R}} + F \right) \right] e^{-2\phi t}. \qquad (4.78)
\end{aligned}
$$

4.8.3.4 Singular Values of the Denominator The denominator $\mathcal{D}_P(t)$ can easily be shown to have a zero for t_0 such that

$$e^{-2\phi t_0} = 1 + 2\frac{R}{H^2} \times \frac{H^2[P(0)\phi + Q] + FR(\phi - F)}{H^2P^2(0) - 2FRP(0) - QR}.$$

However, it can also be shown that $t_0 < 0$ if

$$P(0) > -\frac{R}{H^2}(\phi - F),$$

which is a nonpositive lower bound on the initial value. This poses no particular difficulty, however, since $P(0) \geq 0$ anyway. (We will see in the next section what would happen if this condition were violated.)

4.8.3.5 Boundary Values Given the above formulas for $P(t)$, its numerator $N(t)$, and its denominator $D(t)$, one can easily show that they have the following limiting values:

$$\begin{aligned}
\lim_{t\to 0} N_P(t) &= 2P(0)R\sqrt{F^2 + \frac{H^2Q}{R}}, \\
\lim_{t\to 0} D_P(t) &= 2R\sqrt{F^2 + \frac{H^2Q}{R}}, \\
\lim_{t\to 0} P(t) &= P(0), \\
\lim_{t\to \infty} P(t) &= \frac{R}{H^2}\left(F + \sqrt{F^2 + \frac{H^2Q}{R}}\right).
\end{aligned} \tag{4.79}$$

4.8.4 Parametric Dependence of the Scalar Time-Invariant Solution

The previous solution of the scalar time-invariant problem will now be used to illustrate its dependence on the parameters F, H, R, Q, and $P(0)$. There are two fundamental algebraic functions of these parameters that will be useful in characterizing the behavior of the solutions: the asymptotic solution as $t \to \infty$ and the time constant of decay to this steady-state solution.

4.8.4.1 Decay Time Constant The only time-dependent terms in the expression for $P(t)$ are those involving $e^{-2\phi t}$. The fundamental decay time constant of the solution is then the algebraic function

$$\tau(F, H, R, Q) = 2\sqrt{F^2 + \frac{H^2Q}{R}} \tag{4.80}$$

of the problem parameters. Note that this function does not depend upon the initial value of P.

4.8.4.2 Asymptotic and Steady-State Solutions The asymptotic solution of the scalar time-invariant Riccati differential equation as $t \to \infty$ is given in Equation 4.79. It should be verified that this is also the solution of the corresponding *steady-state* differential equation

$$\dot{P} = 0,$$
$$P^2(\infty)H^2R^{-1} - 2FP(\infty) - Q = 0,$$

which is also called the *algebraic*[10] Riccati equation. This quadratic equation in $P(\infty)$ has two solutions, expressible as algebraic functions of the problem parameters:

$$P(\infty) = \frac{FR \pm \sqrt{H^2QR + F^2R^2}}{H^2}.$$

The two solutions correspond to the two values for the signum ($\pm$). There is no cause for alarm, however. The solution that agrees with Equation 4.79 is the nonnegative one. The other solution is nonpositive. We are only interested in the nonnegative solution, because the variance P of uncertainty is, by definition, nonnegative.

4.8.4.3 Dependence on Initial Conditions For the scalar problem, the initial conditions are parameterized by $P(0)$. The dependence of the solution on its initial value is not continuous everywhere, however. The reason is that there are two solutions to the steady-state equation. The nonnegative solution is *stable* in the sense that initial conditions sufficiently near to it converge to it asymptotically. The nonpositive solution is *unstable* in the sense that infinitesimal perturbations of the initial condition cause the solution to diverge from the nonpositive steady-state solution and converge, instead, to the nonnegative steady-state solution.

4.8.4.4 Convergent and Divergent Solutions The *eventual* convergence of a solution to the nonnegative steady-state value may pass through infinity to get there. That is, the solution may initially *diverge*, depending on the initial values. This type of behavior is shown in Figure 4.3, which is a multiplot of solutions to

[10]So called because it is an algebraic equation, not a differential equation. That is, it is constructed from the operations of algebra, not those of the differential calculus. The term by itself is ambiguous in this usage, however, because there are two entirely different forms of the algebraic Riccati equation. One is derived from the Riccati *differential* equation, and the other is derived from the *discrete-time* Riccati equation. The results are both algebraic equations, but they are significantly different in structure.

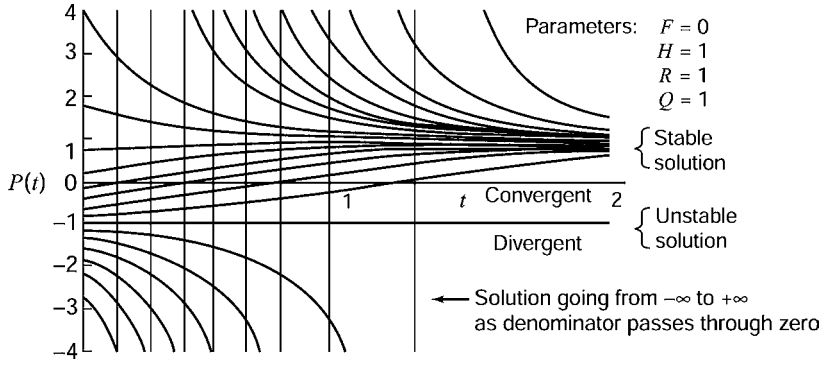


Figure 4.3 Solutions of the scalar time-invariant Riccati equation.

an example of the Riccati equation with

$$F = 0, \quad H = 1, \quad R = 1, \quad Q = 1,$$

for which the corresponding continuous-time algebraic (quadratic) Riccati equation

$$\begin{aligned} \dot{P}(\infty) &= 0, \\ 2FP(\infty) - \frac{[P(\infty)H]^2}{R} + Q &= 0, \\ 1 - [P(\infty)]^2 &= 0. \end{aligned}$$

has the two solutions $P(\infty) = \pm 1$. The Riccati differential equation has the closed-form solution

$$P(t) = \frac{e^{2t}[1 + P(0)] - [1 - P(0)]}{e^{2t}[1 + P(0)] + [1 - P(0)]}$$

in terms of the initial value $P(0)$. Solutions of the initial-value problem with different initial values are plotted over the time interval $0 \leq t \leq 2$. All solutions except the one with $P(0) = -1$ appear to converge eventually to $P(\infty) = 1$, but those that disappear off the bottom of the graph diverge to $-\infty$, then converge to $P(\infty) = +1$ from the top of the graph. The vertical lines on the plot are the times at which solutions starting with $P(0) < -1$ pass through $P(t) = \pm\infty$ on their way to $P(\infty) = 1$. This phenomenon occurs at the zeros of the denominator in the expression for $P(t)$, which occur at time

$$t^* = \log_e \sqrt{\frac{P(0) - 1}{P(0) + 1}}$$

for $P(0) < -1$. Solutions with $P(0) > -1$ converge without this discontinuous behavior.

4.8.4.5 Convergent and Divergent Regions The line at $P = -1$ in Figure 4.3 separates initial values into two regions, characterized by the stability of solutions to the initial-value problem. Solutions with initial values above that line converge to the positive steady-state solution. Solutions starting below that line diverge.

4.8.5 Convergence Issues

It is usually risky to infer properties of high order systems from those of lower order. However, the following general trends are apparent in the behavior of the closed-form solution of the scalar time-invariant Riccati differential equation:

1. The solution eventually converges exponentially to the nonnegative steady-state solution. The decay time constant varies as $(F^2 + H^2\ Q/R)^{1/2}$, which increases with $|F|$, $|H|$, and Q and decreases as R increases (for $R > 0$ and $Q > 0$).
2. Solutions are not uniformly exponentially convergent, however. The initial value does not influence the *asymptotic* decay rate, but it can influence the initial response. In particular, convergence for initial values nearer the unstable steady-state solution is hampered initially.
3. The stable asymptotic solution is

$$P(\infty) = \frac{R}{H^2}\left(F + \sqrt{F^2 + \frac{H^2 Q}{R}}\right),$$

which is influenced by both the sign *and* the magnitude of F but only by the magnitudes of H, R, and Q.

Stability properties of general (higher order) systems have been proved by Potter [215].

4.8.5.1 Even Unstable Dynamic Systems Have Convergent Riccati Equations Note that the corresponding equation for the variance of the *state*

$$\frac{d}{dt}P(t) = FP + PF^{\mathrm{T}} + Q$$

has the general solution

$$P(t) = \frac{(e^{2Ft} - 1)Q}{2F} + e^{2Ft}P(0)$$

in the scalar case. This dynamic system is unstable if $F > 0$, because the solution $P(t) \to +\infty$ as $t \to \infty$. However, the corresponding Riccati equation (with the conditioning term) approaches a finite limit.

4.8.6 Closed-Form Solution of the Algebraic Riccati Equation

We have seen in the previous subsections the difficulty of obtaining a solution of the general Riccati differential equation in closed form (i.e., as a formula in the parameters of the model), even for the simplest (scalar) problem. The following example illustrates the difficulty of obtaining closed-form solutions for the *algebraic* Riccati equation, as well, for a simple model.

Example 4.3 (Solving the Algebraic Riccati Equation in Continuous Time) The problem is to characterize the asymptotic uncertainty in estimating the state (position and velocity) of a damped harmonic resonator driven by Gaussian noise, given noisy measurements of position. The system model for this problem has been derived in Examples 2.2, 2.3, 2.6, 2.7, 3.9, 3.10, and 3.11. The resulting algebraic Riccati equation for this problem in continuous time has the form

$$
\begin{aligned}
0 &= FP + PF^{\mathrm{T}} - PH^{\mathrm{T}}R^{-1}HP + Q, \\
F &= \begin{bmatrix} 0 & 1 \\ -\dfrac{1+\omega^2\tau^2}{\tau^2} & \dfrac{-2}{\tau} \end{bmatrix}, \\
H &= [\,1 \quad 0\,], \\
Q &= \begin{bmatrix} 0 & 0 \\ 0 & q \end{bmatrix},
\end{aligned}
$$

which is equivalent to the three scalar equations

$$
\begin{aligned}
0 &= -p_{11}^2 + 2Rp_{12}, \\
0 &= -R(1+\omega^2\tau^2)p_{11} - 2R\tau p_{12} - \tau^2 p_{11}p_{12} + R\tau^2 p_{22}, \\
0 &= -\tau^2 p_{12}^2 - 2R(1+\omega^2\tau^2)p_{12} - 4R\tau p_{22} + Rq\tau^2.
\end{aligned}
$$

The first and last of these can be solved as linear equations in the variables p_{12} and p_{22}

$$
\begin{aligned}
p_{12} &= \frac{p_{11}^2}{2R}, \\
p_{22} &= \frac{Rq\tau^2 - \tau^2 p_{12}^2 - 2R(1+\omega^2\tau^2)p_{12}}{4R\tau}
\end{aligned}
$$

in terms of p_{11}. Substitution of these expressions into the middle scalar equation yields the following quartic equation in p_{11}:

$$0 = \tau^3 p_{11}^4 + 8R\tau^2 p_{11}^3 + 20R^2\tau(5 + \omega^2\tau^2)p_{11}^2 + 16R^3(1 + \omega^2\tau^2)p_{11} - 4R^3 q\tau^3.$$

This may appear to be a relatively simple quartic equation, but its solution is a rather laborious and tedious process. It has four solutions, only one of which yields a non-negative covariance matrix P:

$$p_{11} = \frac{R(1-b)}{\tau},$$
$$p_{12} = \frac{R(1-b)^2}{2\tau^2},$$
$$p_{22} = \frac{R}{\tau^3}(-6 + 2\omega^2\tau^2 - 4a + (4+a)b),$$
$$a = \sqrt{(1+\omega^2\tau^2)^2 + \frac{q\tau^4}{R}}, \quad b = \sqrt{2(1 - \omega^2\tau^2 + a)}.$$

Because there is no general formula for solving higher order polynomial equations (i.e., beyond quartic), this relatively simple example is at the limit of complexity for finding closed-form solutions to algebraic Riccati equations by purely algebraic means. Beyond this relatively low level of complexity, it is necessary to employ numerical solution methods. Numbers do not always provide as much insight into the characteristics of the solution as formulas do, but they are all we can get for most problems of practical significance.

4.8.7 Newton–Raphson Solution of the Algebraic Riccati Differential Equation

The *Newton–Raphson solution* of n differentiable functional equations

$$\begin{aligned} 0 &= f_1(x_1, x_2, x_3, \ldots, x_n), \\ 0 &= f_2(x_1, x_2, x_3, \ldots, x_n), \\ 0 &= f_3(x_1, x_2, x_3, \ldots, x_n), \\ &\vdots \\ 0 &= f_n(x_1, x_2, x_3, \ldots, x_n) \end{aligned}$$

in n unknowns $x_1, x_2, x_3, \ldots, x_n$ is the iterative vector procedure

$$x \leftarrow x - \mathcal{F}^{-1} f(x) \tag{4.81}$$

using the vector and matrix variables

$$
\begin{aligned}
x &= [\, x_1 \quad x_2 \quad x_3 \quad \cdots \quad x_n \,]^{\mathrm{T}}, \\
f(x) &= [\, f_1(x) \quad f_2(x) \quad f_3(x) \quad \cdots \quad f_n(x) \,]^{\mathrm{T}}, \\
\mathcal{F} &= \begin{bmatrix}
\dfrac{\partial f_1}{\partial x_1} & \dfrac{\partial f_1}{\partial x_2} & \dfrac{\partial f_1}{\partial x_3} & \cdots & \dfrac{\partial f_1}{\partial x_n} \\
\dfrac{\partial f_2}{\partial x_1} & \dfrac{\partial f_2}{\partial x_2} & \dfrac{\partial f_2}{\partial d_3} & \cdots & \dfrac{\partial f_2}{\partial x_n} \\
\dfrac{\partial f_3}{\partial x_1} & \dfrac{\partial f_3}{\partial x_2} & \dfrac{\partial f_3}{\partial x_3} & \cdots & \dfrac{\partial f_3}{\partial x_n} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\dfrac{\partial f_n}{\partial x_1} & \dfrac{\partial f_n}{\partial x_2} & \dfrac{\partial f_n}{\partial x_3} & \cdots & \dfrac{\partial f_n}{\partial x_n}
\end{bmatrix}.
\end{aligned}
$$

Application of this vector-oriented procedure to matrix equations is generally done by "vectorizing" the matrix of unknowns and using Kronecker products to "matricize" F from what would otherwise be four-dimensional data structures. However, the general approach does not take advantage of the symmetry constraints in the matrix Riccati differential equation. There are two such constraints: one on the symmetry of the Riccati equation itself and another on the symmetry of the solution, P. Therefore, in solving the steady-state $n \times n$ matrix Riccati differential equation, there are effectively only $n(n+1)/2$ independent scalar equations in $n(n+1)/2$ scalar unknowns. The $n(n+1)/2$ scalar unknowns can be taken as the upper triangular elements of P, and the $n(n+1)/2$ scalar equations can be taken as those equating upper triangular terms of the matrix equation. We will first describe the equations by which the matrix equation and the matrix unknown can be vectorized, then show the formthat the variables of the Newton–Raphson solution will take for this vectorization.

4.8.7.1 Vectorizing Formulas If one lets the indices i and j stand for the row and column, respectively, of the terms in the matrix Riccati equation, then the respective elements of the *upper triangular* parts of the matrix equation can be vectorized by the single index p, where

$$
\begin{aligned}
&1 \le j \le n, \\
&1 \le i \le j, \\
&p = \tfrac{1}{2} j(j-1) + i, \\
&1 \le p \le \tfrac{1}{2} n(n+1).
\end{aligned}
$$

Similarly, the upper triangular part of P can be mapped into a singly subscripted array x with index q according to the rules

$$
\begin{aligned}
&1 \le \ell \le n,\\
&1 \le k \le \ell,\\
&q = \frac{1}{2}\ell(\ell-1) + k,\\
&1 \le q \le \frac{1}{2}n(n+1),
\end{aligned}
$$

whereby $P_{k\ell}$ is mapped into x_q.

4.8.7.2 Values of Variables for the Newton–Raphson Solution of the Steady-State Matrix Riccati Differential Equation The solution is an implementation of the recursion formula 4.74 with

$$f_p = Z_{ij}, \tag{4.82}$$

$$Z = FP + PF^{\mathrm{T}} - PH^{\mathrm{T}}R^{-1}HP + Q, \tag{4.83}$$

$$x_q = P_{k\ell}, \tag{4.84}$$

$$p = \tfrac{1}{2}\, j(j-1) + i, \tag{4.85}$$

$$q = \tfrac{1}{2}\, \ell(\ell-1) + k, \tag{4.86}$$

$$
\begin{aligned}
\mathcal{F}_{pq} &= \frac{\partial f_p}{\partial x_q}\\
&= \frac{\partial Z_{ij}}{\partial P_{k\ell}}\\
&= \Delta_{j\ell}S_{ik} + \Delta_{ik}S_{j\ell},
\end{aligned} \tag{4.87}
$$

$$S = F - PH^{\mathrm{T}}R^{-1}H, \tag{4.88}$$

$$\Delta_{ab} \overset{\text{def}}{=} \begin{cases} 1 & \text{if} \quad a = b\\ 0 & \text{if} \quad a \neq b. \end{cases} \tag{4.89}$$

The least obvious of these is Equation 4.87, which will now be derived.

4.8.7.3 "Dot" Notation for Row and Column Submatrices For any matrix M, let the notation $M_{\cdot j}$ [with a dot $(\cdot)$ where the row index should be] stand for the jth column of M. When this notation is applied to the identity matrix I, $I_{\cdot j}$ will equal a column vector with 1 in the jth row and 0s elsewhere. As a vector, it has the property that

$$MI_{\cdot j} = M_{\cdot j}$$

for any conformable matrix M.

4.8.7.4 Matrix Partial Derivatives With this notation, one can write matrix partial derivatives as follows:

$$\frac{\partial P}{\partial P_{k\ell}} = I_{\cdot k} I_{\cdot \ell}^{\mathrm{T}}, \tag{4.90}$$

$$\frac{\partial Z}{\partial P_{k\ell}} = F\frac{\partial P}{\partial P_{k\ell}} + \frac{\partial P}{\partial P_{k\ell}}F^{\mathrm{T}} - \frac{\partial P}{\partial P_{k\ell}}H^{\mathrm{T}}R^{-1}HP - PH^{\mathrm{T}}R^{-1}H\frac{\partial P}{\partial P_{k\ell}} \tag{4.91}$$

$$= FI_{\cdot k}I_{\cdot \ell}^{\mathrm{T}} + I_{\cdot k}I_{\cdot \ell}^{\mathrm{T}}F^{\mathrm{T}} - I_{\cdot k}I_{\cdot \ell}^{\mathrm{T}}H^{\mathrm{T}}R^{-1}HP - PH^{\mathrm{T}}R^{-1}HI_k I_{\ell}^{\mathrm{T}} \tag{4.92}$$

$$= F_{\cdot k}I_{\cdot \ell}^{\mathrm{T}} + I_{\cdot k}F_{\cdot \ell}^{\mathrm{T}} - I_{\cdot k}M_{\cdot \ell}^{\mathrm{T}} - M_{\cdot k}I_{\cdot \ell}^{\mathrm{T}} \tag{4.93}$$

$$= (F - M)_{\cdot k}I_{\cdot \ell}^{\mathrm{T}} + I_{\cdot k}(F - M)_{\cdot \ell}^{\mathrm{T}} \tag{4.94}$$

$$= S_{\cdot k}I_{\cdot \ell}^{\mathrm{T}} + I_{\cdot k}S_{\cdot \ell}^{\mathrm{T}}, \tag{4.95}$$

$$S = F - M, \tag{4.96}$$

$$M = PH^{\mathrm{T}}R^{-1}H. \tag{4.97}$$

Note that, on the right-hand side of Equation 4.95, the first term ($S_{\cdot k}\ I_{\cdot \ell}^{\mathrm{T}}$) has only one nonzero column—the ℓth column. Similarly, the other term ($I_{\cdot k}\ S_{\cdot \ell}^{\mathrm{T}}$) has only one nonzero row—its kth row. Consequently, the element in the ith row and jth column of this matrix will be the expression given in Equation 4.87. This completes its derivation.

4.8.7.5 Computational Complexity The number of floating-point operations per iteration for this solution method is dominated by the inversion of the $n(n+1)/2 \times n(n+1)/2$ matrix $\mathcal{F}$, which requires somewhat more than $n^6/8$ flops.

4.8.8 MacFarlane–Potter–Fath Eigenstructure Method

4.8.8.1 Steady-State Solution of the Time-Invariant Matrix Riccati Differential Equation It was discovered independently by MacFarlane [179], Potter [216], and Fath [82] that the solution $P(\infty)$ of the continuous-time form of the steady-state matrix Riccati differential equation can be expressed in the form

$$P(\infty) = AB^{-1},$$

$$\begin{bmatrix} A \\ B \end{bmatrix} = [\, e_{i_1} \quad e_{i_2} \quad e_{i_3} \quad \cdots \quad e_{i_n}\,],$$

where the matrices A and B are $n \times n$ and the $2n$ vectors e_{i_k} are characteristic vectors of the continuous-time system Hamiltonian matrix

$$\Psi_c = \begin{bmatrix} F & Q \\ H^{\mathrm{T}}R^{-1}H & -F^{\mathrm{T}} \end{bmatrix}.$$

This can be formalized in somewhat greater generality as a lemma:

Lemma 1 If A and B are $n \times n$ matrices such that B is nonsingular and

$$\Psi_c \begin{bmatrix} A \\ B \end{bmatrix} = \begin{bmatrix} A \\ B \end{bmatrix} D \tag{4.98}$$

for an $n \times n$ matrix D, then $P = AB^{-1}$ satisfies the steady-state matrix Riccati differential equation

$$0 = FP + PF^{\mathrm{T}} - PH^{\mathrm{T}}R^{-1}HP + Q.$$

Proof Equation 4.98 can be written as two equations,

$$AD = FA + QB, \quad BD = H^{\mathrm{T}}R^{-1}HA - F^{\mathrm{T}}B.$$

If one multiplies both of these on the right by B^{-1} and the last of these on the left by AB^{-1}, one obtains the equivalent equations

$$\begin{aligned} ADB^{-1} &= FAB^{-1} + Q, \\ ADB^{-1} &= AB^{-1}H^{\mathrm{T}}R^{-1}HAB^{-1} - AB^{-1}F^{\mathrm{T}}, \end{aligned}$$

or, taking the differences of the left-hand sides and substituting P for $AB^{-1,}$

$$0 = FP + PF^{\mathrm{T}} - PH^{\mathrm{T}}R^{-1}HP + Q,$$

which was to be proved.

In the case where A and B are formed in this way from n characteristic vectors of Ψ_c, the matrix D will be a diagonal matrix of the corresponding characteristic values. (Check it out for yourself.) Therefore, to obtain the steady-state solution of the matrix Riccati differential equation by this method, it suffices to find n characteristic vectors of Ψ_c such that the corresponding B-matrix is nonsingular. (As will be shown in the next section, the same trick works for the discrete-time matrix Riccati equation.)

4.9 MATRIX RICCATI EQUATION IN DISCRETE TIME

4.9.1 Linear Equations for Matrix Fraction Propagation

The representation of the covariance matrix as a matrix fraction is also sufficient to transform the nonlinear discrete-time Riccati equation for the estimation uncertainty

into a linear form. The discrete-time problem differs from the continuous-time problem in two important aspects:

1. The numerator and denominator matrices will be propagated by a $2n \times 2n$ transition matrix and not by differential equations. The approach is otherwise similar to that for the continuous-time Riccati equation, but the resulting $2n \times 2n$ state transition matrix for the recursive updates of the numerator and denominator matrices is a bit more complicated than the coefficient matrix for the linear form of the continuous-time matrix Riccati equation.
2. There are two distinct values of the discrete-time covariance matrix at any discrete-time step—the *a priori* value and the *a posteriori* value. The *a priori* value is of interest in computing Kalman gains, and the *a posteriori* value is of interest in the analysis of estimation uncertainty.

The linear equations for matrix fraction propagation of the *a priori* covariance matrix are derived below. The method is then applied to obtain a closed-form solution for the scalar time-invariant Riccati equation in discrete time and to a method for exponential speedup of convergence to the asymptotic solution.

4.9.2 Matrix Fraction Propagation of the *a Priori* Covariance

Lemma 2 If the state transition matrices Φ_k are nonsingular and

$$P_k(-) = A_k B_k^{-1} \tag{4.99}$$

is a nonsingular matrix solution of the discrete-time Riccati equation at time t_k, then

$$P_{k+1}(-) = A_{k+1} B_{k+1}^{-1} \tag{4.100}$$

is a solution at time t_{k+1}, where

$$\begin{bmatrix} A_{k+1} \\ B_{k+1} \end{bmatrix} = \begin{bmatrix} Q_k & I \\ I & 0 \end{bmatrix} \begin{bmatrix} \Phi_k^{-\mathrm{T}} & 0 \\ 0 & \Phi_k \end{bmatrix} \begin{bmatrix} H_k^{\mathrm{T}} R_k^{-1} H_k & I \\ I & 0 \end{bmatrix} \begin{bmatrix} A_k \\ B_k \end{bmatrix} \tag{4.101}$$

$$= \begin{bmatrix} \Phi_k + Q_k \Phi_k^{-\mathrm{T}} H_k^{\mathrm{T}} R_k^{-1} H_k & Q_k \Phi_k^{-\mathrm{T}} \\ \Phi_k^{-\mathrm{T}} H_k^{\mathrm{T}} R_k^{-1} H_k & \Phi_k^{-\mathrm{T}} \end{bmatrix} \begin{bmatrix} A_k \\ B_k \end{bmatrix}. \tag{4.102}$$

Proof The following annotated sequence of equalities starts with the product A_{k+1} B_{k+1}^{-1} as defined and proves that it equals P_{k+1}:

$$
\begin{aligned}
A_{k+1}B_{k+1}^{-1} &= \{[\Phi_k + Q_k\Phi_k^{-\mathrm{T}}H_k^{-\mathrm{T}}R_k^{-1}H_k]A_k + Q_k\Phi_k^{-\mathrm{T}}B_k\} \\
&\quad \times \{\Phi_k^{-\mathrm{T}}[H_k^{\mathrm{T}}R_k^{-1}H_kA_kB_k^{-1} + I]B_k\}^{-1} \qquad \text{(definition)} \\
&= \{[\Phi_k + Q_k\Phi_k^{-\mathrm{T}}H_k^{\mathrm{T}}R_k^{-1}H_k]A_k + Q_k\Phi_k^{-\mathrm{T}}B_k\} \\
&\quad \times B_k^{-1}\{H_k^{\mathrm{T}}R_k^{-1}H_kA_kB_k^{-1} + I\}^{-1}\Phi_k^{\mathrm{T}} \qquad \text{(factor } B_k\text{)} \\
&= \{[\Phi_k + Q_k\Phi_k^{-\mathrm{T}}H_k^{\mathrm{T}}R_k^{-\mathrm{T}}H_k]A_kB_k^{-1} + Q_k\Phi_k^{-\mathrm{T}}\} \\
&\quad \times \{H_k^{\mathrm{T}}R_k^{-1}H_kA_kB_k^{-1} + I\}^{-1}\Phi_k^{\mathrm{T}} \qquad \text{(distribute } B_k\text{)} \\
&= \{[\Phi_k + Q_k\Phi_k^{-\mathrm{T}}H_k^{\mathrm{T}}R_k^{-1}H_k]P_k(-) + Q_k\Phi_k^{-\mathrm{T}}\} \\
&\quad \times \{H_k^{\mathrm{T}}R_k^{-1}H_kP_k(-) + I\}^{-1}\Phi_k^{\mathrm{T}} \qquad \text{(definition)} \\
&= \{\Phi_kP_k(-) + Q_k\Phi_k^{-\mathrm{T}}[H_k^{\mathrm{T}}R_k^{-1}H_kP_k(-) + I]\} \\
&\quad \times \{H_k^{\mathrm{T}}R_k^{-\mathrm{T}}H_kP_k(-) + I\}^{-1}\Phi_k^{\mathrm{T}} \qquad \text{(regroup)} \\
&= \Phi_kP_k(-)\{H_k^{\mathrm{T}}R_k^{-\mathrm{T}}H_kP_k(-) + I\}^{-1}\Phi_k^{\mathrm{T}} + Q_k\Phi_k^{-\mathrm{T}}\Phi_k^{\mathrm{T}} \qquad \text{(distribute)} \\
&= \Phi_k\{H_k^{\mathrm{T}}R_k^{-1}H_k + P_k^{-1}(-)\}(-)\Phi_k^{\mathrm{T}} + Q_k \\
&= \Phi_k\{P_k(-) - P_k(-)H_k^{\mathrm{T}}[H_kP_k(-)H_k^{\mathrm{T}} + R_k]^{-1} \\
&\quad \times H_kP_k(-)\}\Phi_k^{\mathrm{T}} + Q_k \qquad \text{(Hemes)} \\
&= P_{k+1}(-), \qquad \text{(Riccati),}
\end{aligned}
$$

where the *Hemes inversion formula* is given in Appendix B. This completes the proof.

This lemma is used below to derive a closed-form solution for the steady-state Riccati equation in the scalar time-invariant case and in Chapter 8 to derive a fast iterative solution method for the matrix time-invariant case.

4.9.3 Closed-Form Solution of the Scalar Time-Invariant Case

Because this case can be solved in closed form, it serves to illustrate the application of the linearization method derived above.

4.9.3.1 Characteristic Values and Vectors The linearization will yield the following 2×2 transition matrix for the numerator and denominator matrices

representing the covariance matrix as a matrix fraction:

$$\Psi = \begin{bmatrix} Q_k & I \\ I & 0 \end{bmatrix} \begin{bmatrix} \Phi_k^{-\mathrm{T}} & 0 \\ 0 & \Phi_k \end{bmatrix} \begin{bmatrix} H_k^{\mathrm{T}} R_k^{-1} H_k & I \\ I & 0 \end{bmatrix}$$
$$= \begin{bmatrix} \Phi + \dfrac{H^2 Q}{\Phi R} & \dfrac{Q}{\Phi} \\ \dfrac{H^2}{\Phi R} & \dfrac{1}{\Phi} \end{bmatrix}.$$

This matrix has characteristic values

$$\lambda_1 = \frac{H^2 Q + R(\Phi^2 + 1) + \sigma}{2\Phi R}, \quad \lambda_2 = \frac{H^2 Q + R(\Phi^2 + 1) - \sigma}{2\Phi R},$$
$$\sigma = \sigma_1 \sigma_2,$$
$$\sigma_1 = \sqrt{H^2 Q + R(\Phi + 1)^2}, \quad \sigma_2 = \sqrt{H^2 Q + R(\Phi - 1)^2},$$

with ratio

$$\rho = \frac{\lambda_2}{\lambda_1}$$
$$= \frac{\psi - [H^2 Q + R(\Phi^2 + 1)]\sigma}{2\Phi^2 R^2}$$
$$\leq 1,$$
$$\psi = [H^2 Q + R(\Phi^2 + 1)]^2 - 2R^2\Phi^2$$
$$= H^4 Q^2 + 2H^2 QR + 2H^2\Phi^2 QR + R^2 + \Phi^4 R^2.$$

The corresponding characteristic vectors are the column vectors of the matrix

$$M = \begin{bmatrix} \dfrac{-2QR}{H^2 QR(\Phi^2 - 1) + \sigma} & \dfrac{-2QR}{H^2 QR(\Phi^2 - 1) - \sigma} \\ 1 & 1 \end{bmatrix},$$

the inverse of which is

$$M^{-1} = \begin{bmatrix} -\dfrac{H^2}{\sigma_2 \sigma_1} & \dfrac{H^2 Q - R + \Phi^2 R + \sigma_2 \sigma_1}{2\sigma_2 \sigma_1} \\ \dfrac{H^2}{\sigma_2 \sigma_1} & \dfrac{-(H^2 Q) + R - \Phi^2 R + \sigma_2 \sigma_1}{2\sigma_2 \sigma_1} \end{bmatrix}$$

$$= \frac{1}{4QR\sigma_1\sigma_2}\begin{bmatrix} \tau_1\tau_2 & 2QR\tau_1 \\ -\tau_1\tau_2 & -2QR\tau_2 \end{bmatrix},$$

$$\tau_1 = H^2Q + R(\Phi^2 - 1) + \sigma, \quad \tau_2 = H^2Q + R(\Phi^2 - 1) - \sigma.$$

4.9.3.2 Closed-Form Solution This will have the form

$$P_k = A_k B_K^{-T}$$

for

$$\begin{bmatrix} A_k \\ B_k \end{bmatrix} = \Psi^k \begin{bmatrix} P_0 \\ 1 \end{bmatrix}$$

$$= M \begin{bmatrix} \lambda_1^k & 0 \\ 0 & \lambda_2^k \end{bmatrix} M^{-1} \begin{bmatrix} P_0 \\ 1 \end{bmatrix}.$$

This can be expressed in the form

$$P_k = \frac{(P_0\tau_2 + 2QR) - (P_0\tau_1 + 2QR)\rho^k}{(2H^2P_0 - \tau_1) - (2H^2P_0 - \tau_2)\rho^k},$$

which is similar in structure to the closed-form solution for the scalar time-invariant Riccati differential equation. In both cases, the solution is a ratio of linear functions of an exponential time function. In the discrete-time case, the discrete-time power ρ^k serves essentially the same function as the exponential function $e^{-2\phi t}$ in the closed-form solution of the differential equation. Unlike the continuous-time solution, however, this discrete-time solution can "skip over" zeros of the denominator.

4.9.4 MacFarlane–Potter–Fath Eigenstructure Method

4.9.4.1 Steady-State Solution of the Time-Invariant Discrete-Time Matrix Riccati Equation The method presented in Section 4.8.8 for the steady-state solution of the time-invariant matrix Riccati differential equation (i.e., in continuous time) also applies to the Riccati equation in discrete time. As before, it is formalized as alemma:

Lemma 3 If A and B are $n \times n$ matrices such that B is nonsingular and

$$\Psi_d \begin{bmatrix} A \\ B \end{bmatrix} = \begin{bmatrix} A \\ B \end{bmatrix} D \tag{4.103}$$

for an $n \times n$ nonsingular matrix D, then $P_\infty = AB^{-1}$ satisfies the steady-state discrete-time matrix Riccati equation

$$P_\infty = \Phi\{P_\infty - P_\infty H^{\mathrm{T}}[HP_\infty H^{\mathrm{T}} + R]^{-1} HP_\infty\}\Phi^{\mathrm{T}} + Q.$$

Proof If $P_k = AB^{-1}$, then it was shown in Lemma 2 that $P_{k+1} = \acute{A}\acute{B}^{-1}$, where

$$\begin{aligned}
\begin{bmatrix} \acute{A} \\ \acute{B} \end{bmatrix} &= \begin{bmatrix} (\Phi_k + Q_k \Phi_k^{-\mathrm{T}} H_k^{\mathrm{T}} R_k^{-1} H_k) & Q_k \Phi_k^{-\mathrm{T}} \\ \Phi_k^{-\mathrm{T}} H_k^{\mathrm{T}} R_k^{-1} H_k & \Phi_k^{-\mathrm{T}} \end{bmatrix} \begin{bmatrix} A \\ B \end{bmatrix} \\
&= \Psi_d \begin{bmatrix} A \\ B \end{bmatrix} \\
&= \begin{bmatrix} A \\ B \end{bmatrix} D \\
&= \begin{bmatrix} AD \\ BD \end{bmatrix}
\end{aligned}$$

Consequently,

$$\begin{aligned}
P_{k+1} &= \acute{A}\acute{B}^{-1} \\
&= (AD)(BD)^{-1} \\
&= ADD^{-1}B^{-1} \\
&= AB^{-1} \\
&= P_k.
\end{aligned}$$

That is, AB^{-1} is a steady-state solution, which was to be proved.

In practice, A and B are formed from n characteristic vectors of Ψ_d. The matrix D will be a diagonal matrix of the corresponding nonzero characteristic values.

4.10 MODEL EQUATIONS FOR TRANSFORMED STATE VARIABLES

The question to be addressed here is, what happens to the Kalman filter model equations when the state variables and measurement variables are redefined by linear transformations? The answer to this question can be derived as a set of formulas, giving the new model equations in terms of the parameters of "old" model equations and the linear transformations relating the two sets of variables. In Chapter 7, these formulas will be used to simplify the model equations.

4.10.1 Linear Transformations of State Variables

These are changes of variables by which the "new" state and measurement variables are linear combinations of the respective old state and measurement variables. Such transformations can be expressed in the form

$$\acute{x}_k = A_k x_k, \tag{4.104}$$

$$\acute{z}_k = B_k z_k \tag{4.105}$$

where x and z are the old variables and $\acute{x}$ and $\acute{z}$ are the new state vector and measurement, respectively.

4.10.1.1 Matrix Constraints One must further assume that for each discrete-time index k, A_k is a *nonsingular* $n \times n$ matrix. The requirements on B_k are less stringent. One need only assume that it is conformable for the product $B_k H_k$, that is, that B_k is a matrix with ℓ columns. The dimension of $\acute{z}_k$ is arbitrary and can depend on k.

4.10.2 New Model Equations

With the above assumptions, the corresponding state, measurement, and state uncertainty covariance equations of the Kalman filter model are transformed into the following forms:

$$\acute{x}_{k+1} = \acute{\Phi}_k \acute{x}_k + \acute{w}_k, \tag{4.106}$$

$$\acute{z} = \acute{H}_k \acute{x}_k + \acute{v}_k, \tag{4.107}$$

$$\acute{P}_k(+) = \acute{P}_k(-) - \acute{P}_k(-)\acute{H}_k^{\mathrm{T}}[\acute{H}_k \acute{P}_k(-)\acute{H}_k^{\mathrm{T}} + \acute{R}_k]\acute{H}_k \acute{P}_k(-), \tag{4.108}$$

$$\acute{P}_{k+1}(-) = \acute{\Phi}_k \acute{P}_k(+)\acute{\Phi}_k^{\mathrm{T}} + \acute{Q}_k, \tag{4.109}$$

where the new model parameters are

$$\acute{\Phi}_k = A_k \Phi_k A_k^{-1}, \tag{4.110}$$

$$\acute{H}_k = B_k H_k A_k^{-1}, \tag{4.111}$$

$$\acute{Q}_k = E\langle \acute{w}_k \acute{w}_k^T \rangle \tag{4.112}$$

$$= A_k Q_k A_k^{\mathrm{T}}, \tag{4.113}$$

$$\acute{R} = E\langle \acute{v}_k \acute{v}_k^{\mathrm{T}} \rangle \tag{4.114}$$

$$= B_k R_k B_k^{\mathrm{T}} \tag{4.115}$$

and the new state estimation uncertainty covariance matrices are

$$\acute{P}_k(\pm) = A_k P_k(\pm) A_k^{\mathrm{T}}.$$

4.11 APPLICATION OF KALMAN FILTERS

Chapter 9 is about applications of Kalman filtering to navigation. The Kalman filter has also been applied to sensor calibration [103], radar tracking [64], manufacturing [256], economics [184], signal processing [256], and freeway traffic modeling [105]—to cite a few examples. This section shows some applications of the programs provided on the companion CD. A simple example of a second-order underdamped oscillator is given here to illustrate the application of the equations in Table 4.3. This harmonic oscillator is an approximation of a longitudinal dynamics of an aircraft short period [39].

Example 4.4 (Resonator Tracking) Consider a linear, underdamped, second-order system with displacement $x_1(t)$, rate $x_2(t)$, damping ratio ζ and (undamped) natural frequency of 5 rad/s, and constant driving term of 12.0 with additive white noise $w(t)$ normally distributed. The second-order continuous-time dynamic equation

$$\ddot{x}_1(t) + 2\zeta\omega\dot{x}_1(t) + \omega^2 x_1(t) = 12 + w(t)$$

can be written in state-space form via the state-space techniques of Chapter 2:

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\omega^2 & -2\zeta\omega \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} w(t) + \begin{bmatrix} 0 \\ 12 \end{bmatrix}.$$

The observation equation is

$$z(t) = x_1(t) + v(t).$$

One can generate 100 data points with plant noise and measurement noise equal to zero with the following initial condition and parameter values:

$$\begin{bmatrix} x_1(0) \\ x_2(0) \end{bmatrix} = \begin{bmatrix} 0 & \text{ft} \\ 0 & \text{ft/s} \end{bmatrix},$$

$$P(0) = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix},$$

$$Q = 4.47(\text{ft/s})^2, \quad R = 0.01(\text{ft})^2,$$

$$\zeta = 0.2, \quad \omega = 5 \text{ rad/s}$$

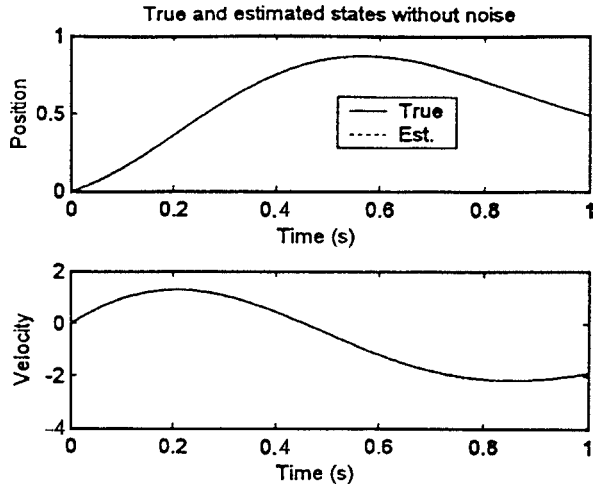


Figure 4.4 Estimated position (ft) and velocity (ft/s) versus time (s).

Equations 4.21, 4.24, 4.25, and 4.26 were programmed in MATLAB on a PC (see Appendix A) to estimate $x_1(t)$ and $x_2(t)$. Figure 4.4 shows the resulting estimates of position and velocity using the noise-free data generated from simulatingthe above second-order equation. Figure 4.5 shows the corresponding RMS uncertainties in position and velocity (top plot), correlation coefficient between position and

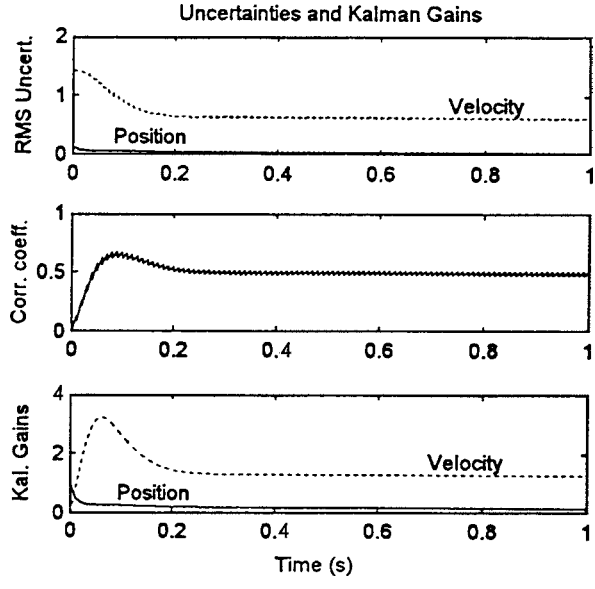


Figure 4.5 RMS uncertainties, position and velocity, correlation coefficient, and Kalman gain.

velocity (middle plot), and Kalman gains (bottom plot). These results were generated from the accompanying MATLAB program `exam43.m` described in Appendix A with sample time = 1 s.

Example 4.5 (Radar Tracking) This example is that of a pulsed *radar tracking system*. In this system, radar pulses are sent out and return signals are processed by the Kalman filter in order to determine the position of maneuvering airborne objects [28]. This example's equations are drawn from [248, 185].

Discrete time dynamic equations in state-space formulation are

$$x_k = \begin{bmatrix} 1 & T & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & \rho & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & T & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & \rho \end{bmatrix} x_{k-1} + \begin{bmatrix} 0 \\ 0 \\ w_{k-1}^1 \\ 0 \\ 0 \\ w_{k-1}^2 \end{bmatrix}.$$

The discrete-time observation equation is given by

$$z_k = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} x_k + \begin{bmatrix} v_k^1 \\ v_k^2 \end{bmatrix},$$

where

$x_k^{\mathrm{T}} = [r_k \quad \dot{r}_k \quad U_k^1 \quad \theta_k \quad \dot{\theta}_k \quad U_k^2]$
$r_k =$ range of the vehicle at time k
$\dot{r}_k =$ range rate of the vehicle at time k
$U_k^1 =$ maneuvering correlated state noise
$\theta_k =$ bearing of the vehicle at time k
$\dot{\theta}_k =$ bearing rate of the vehicle at time k
$U_k^2 =$ maneuveering-correlated state noise
$T =$ sampling period in seconds
$w_k^T = [w_k^1 \quad w_k^2]$ zero-mean white-noise sequences and covariance of σ_1^2 and σ_2^2, respectively
$v_k^T = [v_k^1 \quad v_k^2]$ sensor zero mean white noise sequence and covariance of σ_r^2 and σ_θ^2, respectively, and w_k and v_k are uncorrelated:

$$\rho = \text{correlation coefficient} = \frac{E[U_k U_{k-1}]}{\sigma_m^2} = \begin{cases} 1 - \lambda T & T \le \dfrac{1}{\lambda}, \\ 0 & T > \dfrac{1}{\lambda} \end{cases}$$

where σ_m^2 is the maneuver variance and λ is the inverse of average maneuver duration.

The shaping filter for whitening the maneuver noise is given by

$$U_k^1 = \rho U_{k-1}^1 + w_{k-1}^1,$$

which drives the range rate ($\dot{r}_k$) state of the vehicle, and

$$U_k^2 = \rho U_{k-1}^2 + w_{k-1}^2,$$

which drives the bearing rate (θ_k) state of the vehicle. The derivation of the discrete-time shaping filter is given in Section 3.6 with examples. The range, range rate, bearing, and bearing rate equations have been augmented by the shaping filter equations. The dimension of the state vector is increased from 4×1 to 6×1.

Covariance and gain plots for this system are produced using the Kalman filter program of Appendix A. The following initial covariance (P_0), plant noise (Q), and measurement noise (R) are used to generate the covariance results:

$$P_0 = \begin{bmatrix} \sigma_r^2 & \frac{\sigma_r^2}{T} & 0 & 0 & 0 & 0 \\ \frac{\sigma_r^2}{T} & \frac{2\sigma_r^2}{T^2} + \sigma_1^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_1^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_\theta^2 & \frac{\sigma_\theta^2}{T} & 0 \\ 0 & 0 & 0 & \frac{\sigma_\theta^2}{T} & \frac{2\sigma_\theta^2}{T^2} + \sigma_2^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_2^2 \end{bmatrix}, \quad Q = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_1^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_2^2 \end{bmatrix},$$

$$R = \begin{bmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_\theta^2 \end{bmatrix},$$

Here $\rho = 0.5$ and $T = 5$, 10, and 15 s, respectively. Also,

$$\sigma_r^2 = (1000 \text{ m})^2, \quad \sigma_\theta^2 = (0.017 \text{ rad})^2,$$
$$\sigma_1^2 = (103/3)^2, \quad \sigma_2^2 = 1.3 \times 10^{-8}$$

Parts of this example are discussed in [244]. Results of covariances and Kalman gain plots are shown in Figures 4.6–4.8. Convergence of the diagonal elements of the covariance matrix is shown in these figures for intervals (5, 10, 15 s). Selected Kalman gain values are shown in the following figures for various values of sampling times. These results were generated using the accompanying MATLAB program `exam44.m` described in Appendix A.

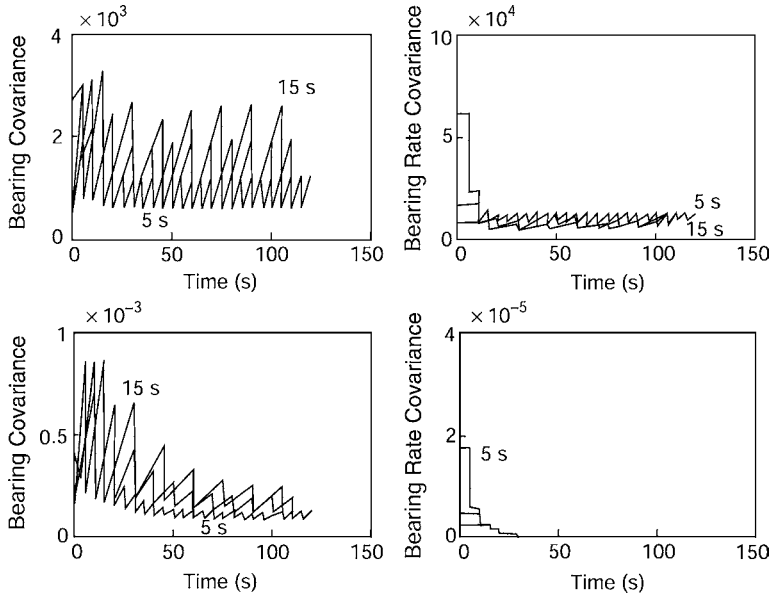


Figure 4.6 Covariances.

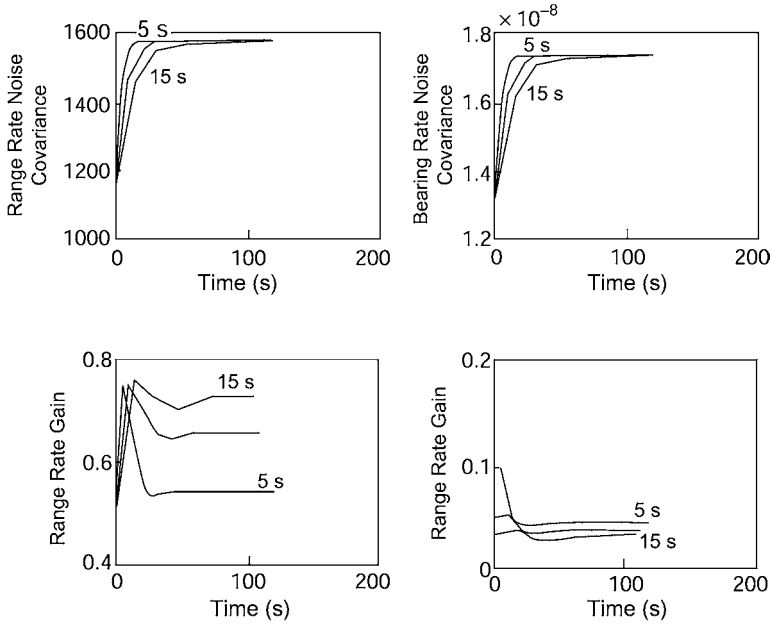


Figure 4.7 Covariances and Kalman gains.

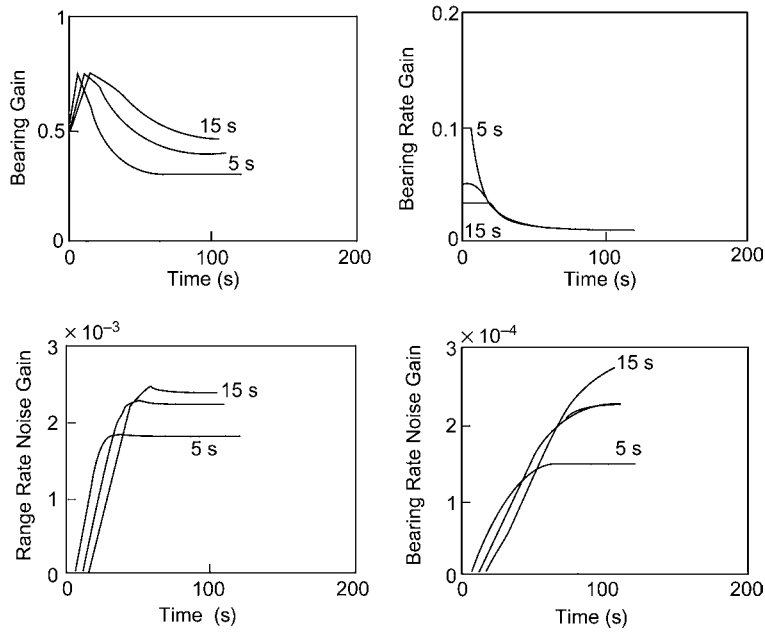


Figure 4.8 Kalman gains.

4.12 SUMMARY

4.12.1 Important Points to Remember

The optimal linear estimator is equivalent to the general (nonlinear) optimal estimator if the random processes x and z are jointly normal. Therefore, the equations for the discrete-time and continuous-time linear optimal estimators can be derived by using the orthogonality principle of Chapter 3. The discrete-time estimator (Kalman filter) has been described, including its implementation equations and block diagram description. The continuous-time estimator (Kalman–Bucy filter) has also been described.

Prediction is equivalent to filtering when measurements (system outputs) are not available. Implementation equations for continuous-time and discrete-time predictors have been given, and the problem of missing data has been discussed in detail. The estimator equations for the case where there is correlation between plant and measurement noise sources and correlated measurement errors were discussed. Relationships between stationary continuous-time and the Kalman and Wiener filters were covered.

Methods for solving matrix Riccati differential equations have been included. Examples discussed include the applications of the Kalman filter to (1) estimating the state (phase and amplitude) of a harmonic oscillator and (2) a discrete-time Kalman filter implementation of a six-dimensional radar tracking problem.

An estimator for the state of a dynamic system at time t, using measurements made *after* time t, is called a *smoother*.

4.12.2 Important Equations to Remember

4.12.2.1 Kalman Filter The *discrete-time model* for a linear stochastic system has the form

$$x_k = \Phi_{k-1}x_{k-1} + G_{k-1}w_{k-1},$$
$$z_k = H_k x_k + v_k,$$

where the zero-mean uncorrelated Gaussian random processes $\{w_k\}$ and $\{v_k\}$ have covariances Q_k and R_k, respectively, at time t_k. The corresponding *Kalman filter equations* have the form

$$\hat{x}_k(-) = \Phi_{k-1}\hat{x}_{k-1}(+),$$
$$P_k(-) = \Phi_{k-1}P_{k-1}(+)\Phi_{k-1}^{\mathrm{T}} + G_{k-1}Q_{k-1}G_{k-1}^{\mathrm{T}},$$
$$\hat{x}_k(+) = \hat{x}_k(-) + \overline{K}_k(z_k - H_k\hat{x}_k(-)),$$
$$\overline{K}_k = P_k(-)H_k^{\mathrm{T}}(H_kP_k(-)H_k^{\mathrm{T}} + R)^{-1},$$
$$P_k(+) = P_k(-) - \bar{K}_kH_kP_k(-),$$

where $(-)$ indicates the *a priori* values of the variables (before the information in the measurement is used) and $(+)$ indicates the *a posteriori* values of the variables (after the information in the measurement is used). The variable $\overline{K}$ is the Kalman gain.

4.12.2.2 Kalman–Bucy Filter The *continuous-time model* for a linear stochastic system has the form

$$\frac{d}{dt}x(t) = F(t)x(t) + G(t)w(t),$$
$$z(t) = H(t)x(t) + v(t),$$

where the zero-mean uncorrelated Gaussian random processes $\{w(t)\}$ and $\{v(t)\}$ have covariances $Q(t)$ and $R(t)$, respectively, at time t. The corresponding *Kalman–Bucy filter* equations for the estimate $\hat{x}$ of the state variable x, given the output signal z, have the form

$$\frac{d}{dt}\hat{x}(t) = F(t)\hat{x}(t) + \overline{K}(t)[z(t) - H(t)\hat{x}(t)],$$
$$\overline{K}(t) = P(t)H^{\mathrm{T}}(t)R^{-1}(t),$$
$$\frac{d}{dt}P(t) = F(t)P(t) + P(t)F^{\mathrm{T}}(t) - \overline{K}(t)R(t)\overline{K}^{\mathrm{T}}(t) + G(t)Q(t)G^{\mathrm{T}}(t).$$

PROBLEMS

4.1 A scalar discrete-time random sequence x_k is given by

$$x_{k+1} = 0.5x_k + w_k,$$
$$Ex_0 = 0,\ Ex_0^2 = 1,\ Ew_k^2 = 1,\ Ew_k = 0,$$

where w_k is white noise. The observation equation is given by

$$z_k = x_k + v_k$$

$Ev_k = 0$, $Ev_k^2 = 1$, and v_k is also white noise. The terms x_0, w_k, and v_k are all Gaussian. Derive a (nonrecursive) expression for

$$E[x_1 | z_0, z_1, z_2]$$

4.2 For the system given in Problem 4.1:

(**a**) Write the discrete-time Kalman filter equations.

(**b**) Provide the correction necessary if z_2 was not received.

(**c**) Derive the loss in terms of the estimate $\hat{x}_3$ due to missing z_2.

(**d**) Derive the filter for $k \to \infty$ (steady state).

(**e**) Repeat (d) when every other observation is missed.

4.3 In a single-dimension example of a radar tracking an object by means of track-while-scan, measurements of the continuous-time target trajectory at some discrete times are made. The process and measurement models are given by

$$\dot{x}(t) = -0.5x(t) + w(t), \quad z_{kT} = x_{kT} + v_{kT},$$

where T is the intersampling interval (assume 1 s for simplicity):

$$Ev_k = Ew(t) = 0,$$
$$Ew(t_1)\, w(t_2) = 1\delta(t_2 - t_1),$$
$$Ev_{k_1\mathrm{T}} v_{k_2\mathrm{T}} = 1\Delta(k_2 - k_1),$$
$$E\langle v_k w^{\mathrm{T}} \rangle = 0.$$

Derive the minimum mean-squared-filter of $x(t)$ for all t.

4.4 In Problem 4.3, the measurements are received at discrete times and each measurement is spread over some nonzero time interval (radar beam width

nonzero). The measurement equation of Problem 4.3 can be modified to

$$z_{kT+\eta} = x_{kT+\eta} + v_{kT+\eta},$$

where

$$k = 0, 1, 2, \ldots, 0 \leq \eta \leq \eta_0.$$

Let $T = 1$ s, $\eta_0 = 0.1$ (radar beam width) and $v(t)$ be a zero-mean white Gaussian process with covariance equal to 1. Derive the minimum mean-squared filter of $x(t)$ for all t.

4.5 Prove the condition in the discussion following Equation 4.9 that $Ew_k z_i^{\mathrm{T}} = 0$ for $i = 1, \ldots, k$ when w_k and v_k are uncorrelated and white.

4.6 In Example 4.4, use white noise as a driving input to range rate ($\dot{r}_k$) and bearing rate ($\dot{\theta}_k$) equations instead of colored noise. This reduces the dimension of the state vector from 6×1 to 4×1. Formulate the new observation equation. Generate the covariance and Kalman gain plots for the same values of P_0, Q, R, σ_r^2, $\sigma_{\dot{\theta}}^2$, σ_1^2, and σ_2^2.

4.7 For the same problem as Problem 4.6, obtain values of the plant covariance Q for the four-state model such that the associated mean-squared estimation uncertainties for range, range rate, bearing, and bearing rate are within 5–10% of those for the six-state model. (*Hint:* This should be possible because the plant noise is used to model the effects of linearization errors, discretization errors, and other unmodeled effects or approximations. This type of suboptimal filtering will be discussed further in Chapter 8.)

4.8 For the estimation problem modeled by the equations

$$\begin{aligned} x_k &= x_{k-1} + w_{k-1}, \\ w_k &\sim \mathcal{N}(0,30) \text{ and white}, \\ z_k &= x_k + v_k, \\ v_k &\sim \mathcal{N}(0,20) \text{ and white}, \\ P_0 &= 150, \end{aligned}$$

calculate the values of $P_k(+)$, $P_k(-)$, and $\overline{K}_k$ for $k = 1, 2, 3, 4$ and $P_\infty(+)$ (the steady-state value).

4.9 Parameter estimation problem. Let x be a zero-mean Gaussian random variable with covariance P_0, and let $z_k = x + v_k$ be an observation of x with noise $v_k \sim \mathcal{N}(0, R)$.

(**a**) Write the recursion equations for $P_k(+)$, $P_k(-)$, $\overline{K}_k$, and $\hat{x}_k$.

(**b**) What is the value of x_1 if $R = 0$?

(**c**) What is the value of x_1 if $R = +\infty$?

(**d**) Explain the results of (**b**) and (**c**) in terms of measurement uncertainty.

4.10 Assume a stochastic system in continuous time modeled by the equations

$$\begin{aligned}\dot{x}(t) &= -x(t) + w(t),\\ w(t) &\sim \mathcal{N}(0, 30),\\ z(t) &= x(t) + v(t),\\ v(t) &\sim \mathcal{N}(0{,}20).\end{aligned}$$

(**a**) Derive the values of the mean-squared estimation error $P(t)$ and Kalman gain $\overline{K}(t)$ for time $t = 1, 2, 3, 4$.

(**b**) Solve for the steady-state value of P.

4.11 Show that the matrices P_k and $P(t)$ of Equations 4.23 and 4.37 are *symmetric*. That is, $P_k^{\mathrm{T}} = P_k$ and $P^{\mathrm{T}}(t) = P(t)$.

4.12 Derive the observability matrices for Example 4.4 and Problem 4.6 and determine whether these systems are observable.

4.13 A vector discrete-time random sequence x_k is given by

$$x_k = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x_{k-1} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} w_{k-1},$$

$$w_k \sim \mathcal{N}(0{,}1) \text{ and white.}$$

The observation equation is given by

$$z_k = [1 \quad 0] x_k + v_k,$$

$$v_k \sim \mathcal{N}[0, 2 + (-)^k] \text{ and white.}$$

Calculate the values of $P_k(+)$, $P_k(-)$, and $\overline{K}_k$ for $k = 1, \ldots, 10$ and $P_\infty(+)$ (the steady-state value) with

$$P_0 = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}.$$

4.14 Calculate the values of $P_k(+)$, $\overline{K}_k$, $\hat{x}_k(+)$ by serial processing of vector measurement:

$$\hat{x}_k(-) = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, P_k(-) = \begin{bmatrix} 4 & 1 \\ 1 & 9 \end{bmatrix}, H_k = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, R_k = \begin{bmatrix} 3 & 0 \\ 0 & 4 \end{bmatrix}, Z_k = \begin{bmatrix} 3 \\ 4 \end{bmatrix}.$$

5

OPTIMAL SMOOTHERS

> Many difficulties which nature throws in our way may be smoothed away by the exercise of intelligence.
>
> —Livy [Titus Livius] (59 B.C.E.–17 C.E.)

5.1 CHAPTER FOCUS

5.1.1 Smoothing and Smoothers

What We Mean by Optimal Smoothing The term *smoothing* has many meanings in different contexts, dating from antiquity. *Optimal smoothing* has been around at least since 1795, when Gauss discovered the method of least squares. It later became a part of Wiener–Kolmogorov filtering theory. Since the 1960s, most of the methods used in optimal smoothing have come from Kalman and Kalman–Bucy filtering theory.

Modern optimal smoothing methods are derived from the same models as the Kalman filter for solving the same sort of estimation problem—but not necessarily in real time. The Kalman filter optimizes the use of all measurements made *at or before* the time that the estimated state vector is valid. Smoothing can do a better job than the Kalman filter by using additional measurements made *after* the time of the estimated state vector.

The term *linear smoothing* is also used to indicate that linear dynamic models of the underlying processes are used as part of the smoother, but this terminology could

Kalman Filtering: Theory and Practice using MATLAB®, Third Edition. By Mohinder S. Grewal and Angus P. Andrews

be interpreted to mean that some sort of straight-line fitting is used. In fact, there are many smoothing methods based on least-squares fitting of locally smooth functions (e.g., polynomial splines) to data—but these methods are not included here. The emphasis here is on extensions of Kalman filtering that use the same linear stochastic systems as models for the underlying dynamic processes but relax the relative timing constraints on estimates and measurements.

Smoothers are the algorithmic or analog implementations of smoothing methods. If the application has input signal frequencies too high for sampling or computation rates too fast for digital processing, then the analogous Kalman–Bucy theory in continuous time can be used for designing the implementation circuitry—in much the same way as it was done using Wiener filtering theory.

5.1.2 Kalman Filtering, Prediction, Interpolation, and Smoothing

Kalman Filtering ***Theory*** Kalman filtering theory was developed in Chapters 2 through 4. The same theoretical models have been used in developing optimal smoothing methods.

Prediction Prediction was also covered in Chapter 4. It is an integral part of Kalman filtering, because the estimated values and their associated covariances of uncertainty are always predicted ahead one time step as *a priori* variables. In practice, this sort of prediction is generally continued forward whenever anticipated measurements are unavailable or otherwise deemed unreliable.

Interpolation Interpolation usually refers to estimating state variables during periods of time when no measurements are available, using measurements from adjoining time segments to fill in the "data gaps." Optimal smoothers are also used for interpolation, even though they are usually defined for estimating state variables at those times when measurements are available.

For example, the Kalman filters in some global navigation satellite system (GNSS) receivers continue forward prediction of position and velocity estimates whenever the signals from satellites are unavailable. This is pure prediction, and it is done to aid in rapid reacquisition of signals when they next become available. Once the signals are again available, however, optimal smoothing can also be used to improve (after the fact) the predicted navigation estimates made during the signal outage. *That* is interpolation.

Smoothing Smoothing is the subject of this chapter. Its relationship to optimal predictors and optimal filters is illustrated in Figure 5.1, the distinction depending on when an estimated value of the state vector $\hat{\mathbf{x}}$ is valid and how that time is related to the times at which those measurements used in making that estimate were sampled.

- *Filtering* occupies the diagonal line in the figure, where each state vector estimate is based on those measurements made up to and including the time when the estimate is valid.

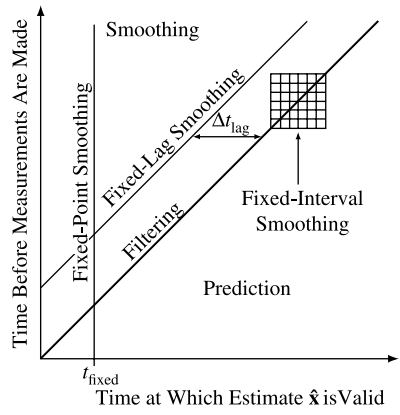


Figure 5.1 Estimate/measurement timing constraints.

- *Prediction* occupies the region below that diagonal line. Each predicted estimate is based on measurement data taken strictly *before* the time that estimate is valid.
- *Smoothing* occupies the region above the diagonal. Each smoothed estimate is based on measurement data taken both *before and after* the time that estimate is valid. The three different types of smoothers labeled in Figure 5.1 are briefly describe in the following subsection.

5.1.3 Types of Smoothers

Most applications of smoothing have evolved into three distinct types, according to the measurement data dependencies of the estimated state vector, as illustrated in Figure 5.1. The following descriptions spell out the constraints on the times measurements are made relative to the times that the value of state vector is to be estimated and provide some examples of how the different smoothers have been used in practice:

5.1.3.1 Fixed-Interval Smoothers Fixed-interval smoothers use all the measurements made at times t_{meas} over a fixed time interval $t_{start} \leq t_{meas} \leq t_{end}$ to produce an estimated state vector $\hat{\mathbf{x}}(t_{est})$ at times $t_{start} \leq t_{est} \leq t_{end}$ in the same fixed interval—as illustrated by the square area along the diagonal in Figure 5.1.

The time at which fixed-interval smoothing is done can be any time after all these measurements have been taken. The most common use of fixed-interval smoothing is for postprocessing of all measurements taken during a test procedure. Fixed-interval smoothing is generally not a real-time process, because the information processing generally takes place after measurements ceased.

Technical descriptions, derivations, and performance analysis of fixed-interval smoothers are presented in Section 5.2.

5.1.3.2 Fixed-Lag Smoothers Fixed-lag smoothers use all measurements made over a time interval $t_{start} \leq t_{meas} \leq t_{est} + \Delta t_{lag}$ for the estimate $\hat{\mathbf{x}}(t_{est})$ at time t_{est}. That is, the estimate generated at time t is for the value of $\mathbf{x}$ at time $t - \Delta t_{lag}$, where Δt_{lag} is a fixed lag time, as illustrated by the sloped line above the "Filtering" line in Figure 5.1.

Fixed-lag smoothers are used in communications for improving signal estimation. They sacrifice some delay in order to reduce bit error rates. Fixed-lag smoothers run in real time, using all measurement up to the current time, but generate an estimate in lagged time.

Technical descriptions, derivations and performance analysis of fixed-lag smoothers are presented in Section 5.3.

5.1.3.3 Fixed-Point Smoothers Fixed-point smoothers generate an estimate $\hat{\mathbf{x}}(t_{fixed})$ of $\mathbf{x}$ at a fixed time t_{fixed} based on all measurements $\mathbf{z}(t_{meas})$ up to the current time t (i.e., $t_{start} \leq t_{meas} \leq t$). These are illustrated by the vertical line in Figure 5.1. Fixed-point smoothers function as predictors when the current time $t < t_{fixed}$, as filters when $t = t_{fixed}$, and as smoothers when $t > t_{fixed}$.

Fixed-point smoothing is useful for estimation problems in which the system state is only of interest at some epochal event time t_{fixed}, which is often the initial state. It is used for initial alignment of inertial navigation systems (INS) [104] and for estimating the initial INS attitudes on vehicles using GNSS receivers as auxiliary navigation sensors. GNSS receivers provide no attitude information during initial INS alignment, but they can detect subsequent patterns of position errors from which initial attitude errors can be estimated using fixed-point smoothing.

Technical descriptions, derivations, and performance analysis of fixed-point smoothers are presented in Section 5.4.

5.1.4 Implementation Algorithms

We describe in this chapter the different types of optimal smoothers, their implementation algorithms, some example applications, and some indication of their relative advantage over optimal filters.

Many different smoothing algorithms have been derived and published, but not all turned out to be practical. Important criteria for smoothing algorithms are

1. Numerical stability: Some early algorithms were numerically unstable.
2. Computational complexity, which limits the data rates at which they can operate.
3. Memory requirements.

Some smoother implementations are in continuous time.

5.1.5 Smoothing Applications

In practice, smoothing is often overlooked as a practical method for getting better estimates than those attainable by filtering alone. Depending on the attributes of

the application, the improvement of smoothing over filtering can be many orders of magnitude in mean-squared estimation uncertainty.

5.1.5.1 Determining Whether and How to Use Smoothing Questions you should consider when deciding whether to use smoothing—and which type of smoothing to use—include the following:

1. What are the constraints imposed by the application on the relative times when the measurements are taken, when the resulting estimates of the system state vector should be valid, and when the results are needed?

   (a) If the estimated state vector values over an entire interval are required, then fixed-interval smoothing should be considered—especially if the results are not needed in near-real time.
   (b) If the estimated state vector value is required to be valid only at one instant of time, then fixed-point smoothing is an appropriate estimation method.
   (c) If results are required in near-real time (i.e., with limited delays), then fixed-lag smoothing may be appropriate.

2. What is the required, expected, or hoped-for improvement (over filtering) of uncertainty in the estimated values of state variables for the intended application?
3. How might the reliability of results be affected by numerical stability of the estimation algorithm or by over-sensitivity of performance to assumed model parameter values?
4. How might computational complexity compromise the cost and attainability of implementation? This issue is more important for real-time applications of fixed-lag and fixed-point smoothing. Smoothing uses more data for each estimated value, and this generally requires more computation than for filtering. One must always consider the computational resources required, whether they are attainable, and their cost.
5. Are there any serious memory storage limitations? This could be a problem for some applications of fixed-interval smoothing with very large data sets.

5.1.6 Improvement Over Filtering

Theoretical limits on the asymptotic improvement of smoothing over filtering were published by B. D. O. Anderson in 1969 [5] for the case in which the dynamic system is asymptotically exponentially stable. The limit in that case is a factor of 2 in mean-squared estimation uncertainty, but greater improvement is attainable for unstable systems.

Relative Improvement For *scalar* models (i.e., $n = 1$), the relative improvement of smoothers over filters can be characterized by the ratios of their respective variances of uncertainty in the resulting estimates.

For multidimensional problems (i.e., $n > 1$) with $P_{[s]}$ as the covariance matrix of smoothing uncertainty and $P_{[f]}$ as the covariance matrix of filtering uncertainty, we would say that smoothing is an improvement over filtering if

$$P_{[s]} < P_{[f]}, \text{ or } [P_{[f]} - P_{[s]} \text{ is positive-definite}].$$

This can generally be determined by implementing the smoother and filter and comparing their covariance matrices of estimation uncertainty.

Dependence on Model Parameters In order to gain some insight into the relative benefits of smoothing and how they depend on attributes of the application, we develop in Sections 5.2 through 5.4 performance analysis models to evaluate how the improvement of smoothing over filtering depends on the parameters of the model. All these models are for linear time-invariant problems, and the models for which we can plot the dependence of performance on problem parameters are all for scalar[1] state variables (i.e., $n = 1$). Even for these scalar models, depending on the parameters of the model, the relative improvement of smoothing over filtering can range from hardly anything to orders of magnitude reduction in mean-squared uncertainty.

5.1.6.1 The Anderson–Chirarattananon Bound This result, due to Brian D. O. Anderson and Surapong Chirarattananon [7], was derived in the frequency domain and uses the signal-to-noise ratio (SNR) as a parameter. This supposes that the dynamic process model representing the signal is *stable* in order that the signal have finite steady-state power.

In Kalman filter models, mean-squared noise is represented by the parameter R, the covariance of measurement noise, and the *signal* by the noise-free part of the measurement $z(t)$ (i.e., Hx). This requires that the dynamic process representing the component of the state vector x in the rank space of H be stable, which implies that the corresponding dynamic coefficient matrix F have negative characteristic values. Sensor noise is assumed to be white, which has constant power spectral density (PSD) across all frequencies. The peak SNR then occurs at that frequency where the signal power spectral density peaks.

The result, shown in Figure 5.2 in two forms, is a bound on the improvement of smoothing over filtering, defined by the ratio of smoothing error variance to filtering error variance. The curve in Figure 5.2(*a*) is a lower bound on the improvement ratio; that in Figure 5.2(*b*) is an upper bound on the equivalent percent improvement in mean-squared signal estimation error from using smoothing instead of filtering.

Additional bounds are derived in this chapter for the relative improvement of each type of smoothing over filtering in terms of how the improvement depends on the values of the underlying linear stochastic system parameters.

[1]Setting $n = 1$ has more to do with dimensionality of the independent parameters (for plotting) than with the feasibility of the performance analysis.

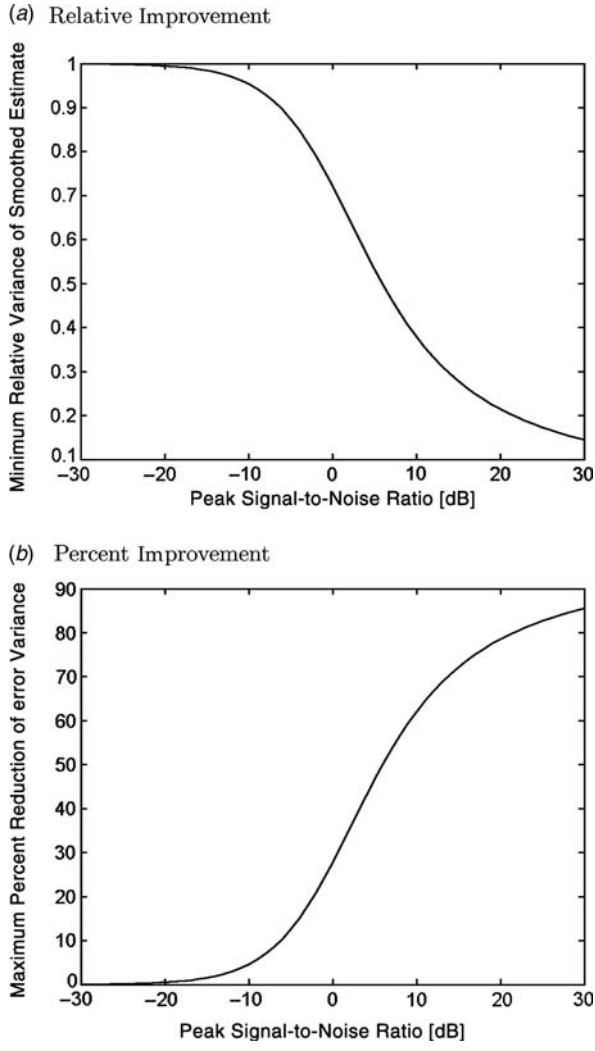


Figure 5.2 Anderson–Chirarattananon bound on smoothing variance improvement over filtering.

5.2 FIXED-INTERVAL SMOOTHING

5.2.1 Performance Analysis in Continuous Time

5.2.1.1 Scalar Parametric Models

Linear Time-Invariant Models in Continuous Time Smoother performance models are developed in continuous time for pedagogical reasons: the resulting model equations are more easily solved in closed form. Smoother performance can also

be determined for the discrete-time models and for higher-dimensional state vectors—although solution may require numerical methods.

Performance Formulas We present some general methods for determining how the relative performance of smoothing with respect to filtering depends on the time-invariant parameters F, Q, H, and R of linear models. These methods are then demonstrated for the scalar state variable case, although the general approach works for arbitrary state vector and measurement vector dimensions.

Stochastic System Model The parametric stochastic system model is

$$\left.\begin{array}{l} \dot{x} = Fx + w \text{ (dynamic system model)} \\ z = Hx + v \text{ (measurement model)} \\ \dot{P} = FP + PF - PHR^{-1}H^{\mathrm{T}}P + Q \text{ (Riccati differential equation).} \end{array}\right\} \tag{5.1}$$

Parameter Units for n = *1* When the state vector has dimension $n = 1$, the scalar Riccati equation variables and parameters have the following units:

$$\left.\begin{array}{ll} P & \text{has the same units as } x^2. \\ F & \text{has units of } 1/\text{time (e.g., } \mathrm{s}^{-1}). \\ H & \text{has } z \text{ (measurement) units divided by } x \text{ (state variable) units.} \\ R & \text{has the units of time} \times z^2 \text{ (measurement units)}^2. \\ Q & \text{has the units of } x^2/\text{time.} \end{array}\right\} \tag{5.2}$$

The seemingly strange units of R and Q are a consequence of stochastic integration using the stochastic differential equation model.

Two-Filter Model For fixed-interval smoothing, performance models are based on two Kalman–Bucy filters:

- A forward filter running forward in time. At each instant of time, the estimate from the forward filter is based on all the measurements made up to that time, and the associated estimation uncertainty covariance characterizes estimation uncertainty based on all those measurements.
- A backward filter running backward in time. At each instant of time, the estimate from the backward filter is based on all the measurements made after that time, and the associated estimation uncertainty covariance characterizes estimation uncertainty based on all those measurements. When time is reversed, the sign on the dynamic coefficient matrix F in Equation 5.1 changes, which can make the performance of the backward filter model different from that of the forward filter model. Th covariance $P_{[b]}$ of backward filter uncertainty can be different from the covariance $P_{[f]}$ of forward filter uncertainty.

Why the Two-Filter Model? At each time t, the forward filter generates the covariance matrix $P_{[f]}(t)$ representing the mean-squared uncertainty in the estimate $\hat{x}_{[f]}(t)$ using all measurements $z(s)$ for $s \leq t$. Similarly, the backward filter generates the covariance matrix $P_{[b]}(t)$ representing the mean-squared uncertainty in the estimate $\hat{x}_{[b]}(t)$ using all measurements $z(s)$ for $s \geq t$. The optimal smoother combines $\hat{x}_{[f]}(t)$ and $\hat{x}_{[b]}(t)$, using $P_{[f]}(t)$ and $P_{[b]}(t)$ in a Kalman filter to minimize the resulting covariance matrix $P_{[s]}(t)$ of smoother uncertainty. $P_{[s]}(t)$ will tell us how well the smoother performs.

Smoother Estimate and Uncertainty At each instant of time, the smoother estimate is obtained by optimally combining two (forward and backward) estimates. Estimates obtained this way will be based on all the measurements, before and after the time of the estimate, over the entire fixed interval. Let

$\hat{x}_{[f]}(t)$ be the forward filter estimate at time t,
$P_{[f]}(t)$ be the associated covariance of estimation uncertainty,
$\hat{x}_{[b]}(t)$ be the backward filter estimate at time t, and
$P_{[b]}(t)$ be the associated covariance of estimation uncertainty.

If we treat $\hat{x}_{[f]}(t)$ as the *a priori* smoother estimate and $\hat{x}_{[b]}(t)$ as an independent measurement, then the optimal smoothed estimate of x at time t will be

$$\hat{x}_{[s]}(t) = \hat{x}_{[f]}(t) + \underbrace{P_{[f]}(t)[P_{[f]}(t) + P_{[b]}(t)]^{-1}}_{\bar{K}} [\hat{x}_{[b]}(t) - \hat{x}_{[f]}(t)],$$

which is essentially the Kalman filter observational update formula with

$$\begin{aligned}
\hat{x}_{[f]}(t) &= \text{the smoother } \textit{a priori} \text{ estimate} \\
H &= I \text{ (identity matrix)} \\
z(t) &= \hat{x}_{[b]}(t) \text{ (backward filter estimate)} \\
R &= P_{[b]}(t) \\
\bar{K} &= P_{[f]}(t)[P_{[f]}(t) + P_{[b]}(t)]^{-1} \text{ (Kalman gain)} \\
\hat{x} + [s](t) &= \text{the smoother } \textit{a posteriori} \text{ estimate} \\
P_{[s]}(t) &= \text{the } \textit{a posteriori} \text{ covariance of smoother estimation uncertainty} \\
&= P_{[f]}(t) - P_{[f]}(t)[P_{[f]}(t) + P_{[b]}(t)]^{-1} P_{[f]}(t) \\
&= P_{[f]}(t)\{I - [P_{[f]}(t) + P_{[b]}(t)]^{-1} P_{[f]}(t)\} \\
&= P_{[f]}(t)[P_{[f]}(t) + P_{[b]}(t)]^{-1}\{[P_{[f]}(t) + P_{[b]}(t)] - P_{[f]}(t)\} \\
&= P_{[f]}(t)[P_{[f]}(t) + P_{[b]}(t)]^{-1} P_{[b]}(t) \qquad (5.3)
\end{aligned}$$

The last of these equations characterizes smoother covariance as a function of forward and backward filter covariances, which are the solutions of appropriate Riccati equations.

5.2.1.2 Infinite-Interval Case This is the easiest model for solving for smoother performance. It generally requires solving two algebraic Riccati equations, but this can be done in closed form for the continuous-time model with a scalar state vector.

General Linear Time-Invariant Model In the linear time-invariant case on the interval $-\infty < t < +\infty$, the Riccati equations for the forward and backward Kalman–Bucy filters will be at steady state:

$$\begin{aligned} 0 &= \frac{d}{dt}P_{[f]}(t) \\ &= FP_{[f]} + P_{[f]}F^{\mathrm{T}} + Q - P_{[f]}H^{\mathrm{T}}R^{-1}HP_{[f]} \\ 0 &= \frac{d}{dt}P_{[b]}(t) \\ &= -FP_{[b]} - P_{[b]}F^{\mathrm{T}} + Q - P_{[b]}H^{\mathrm{T}}R^{-1}HP_{[b]}, \end{aligned}$$

where the backward filter model has the sign on the dynamic coefficient matrix F reversed. In general, these algebraic Riccati equations can be solved by numerical methods and inserted into Equation 5.3 to determine smoother performance.

Scalar State Vector Case This reduces to two quadratic equations, both of which can be solved in closed form for the single positive scalar solution:

$$P_{[f]} = \sqrt{\left(\frac{FR}{H^2}\right)^2 + \left(\frac{QR}{H^2}\right)} + \left(\frac{FR}{H^2}\right) \tag{5.4}$$

$$P_{[b]} = \sqrt{\left(\frac{FR}{H^2}\right)^2 + \left(\frac{QR}{H^2}\right)} - \left(\frac{FR}{H^2}\right) \tag{5.5}$$

$$P_{[s]} = \frac{\left(\frac{QR}{H^2}\right)}{2\sqrt{\left(\frac{FR}{H^2}\right)^2 + \left(\frac{QR}{H^2}\right)}} \tag{5.6}$$

All these formulas depend on just two parameter expressions:

$$\left(\frac{FR}{H^2}\right) \quad \text{and} \quad \left(\frac{QR}{H^2}\right),$$

meaning that mean-squared filter and smoother performance can be plotted as a function of these two parameter expressions.

Results Figures 5.3 and 5.4 are plots of infinite-interval smoothing performance relative to filtering performance as functions of the parameter expressions FR/H^2 and QR/H^2, using the formulas derived above.

Influence of the Sign of F on Relative Performance The sign of F makes a big difference in the benefits of smoothing over filtering. The parameters R, Q, and H^2

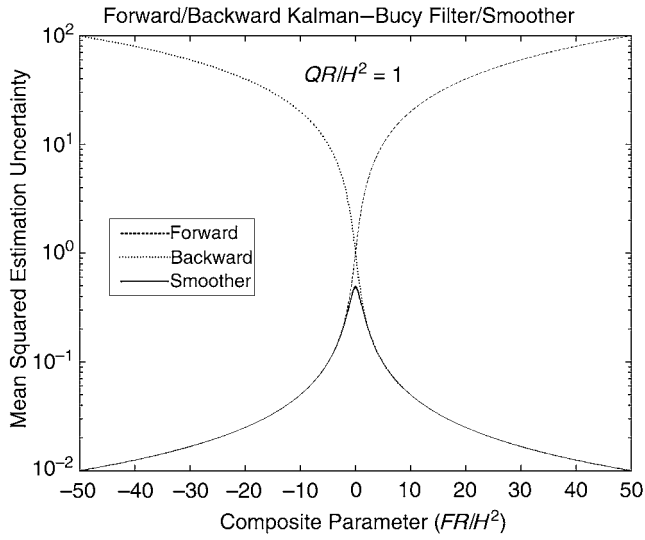


Figure 5.3 Dependence of Kalman–Bucy filter and smoother performance on FR/H^2.

are all positive, but F can change sign. The influence of the sign and magnitude of F on filtering and smoothing performance is plotted in Figure 5.3, which shows the steady-state variance of forward and backward filtering and smoothing as a function of FR/H^2, with the other parameter $QR/H^2 = 1$. The forward filter is stable for $F < 0$ and the backward filter is stable for $F > 0$. For both filters, performance in the stable direction is quite close to smoother performance (solid line), but filter performance in the unstable direction is much worse than smoothing. In essence, smoothing provides a significant improvement over filtering when the backward filter performance is much better than the forward filter performance. For the range of values of FR/H^2 shown (± 50), the variance of smoothing uncertainty is as much as 10,000 times smaller than that for filtering.

Influence of FR/H^2 and QR/H^2 Over Large Dynamic Ranges This is shown with more detail in Figures 5.4(*a*) and (*b*), which are three-dimensional log-log-log plots of the *smoothing improvement ratio* $P_{[s]}/P_{[f]}$ as functions of both composite parameters, (FR/H^2) and (QR/H^2), ranging over ± 4 orders of magnitude.

Unstable Systems For $F > 0$ (Figure 5.4*a*), the smoothing improvement ratio $P_{[s]}/P_{[f]} \to 0$ (arbitrarily better smoothing than filtering) as $(FR/H^2) \to -\infty$, due principally to degrading filter performance, not improving smoother performance. The improvement of smoothing over filtering also improves significantly as $(QR/H^2) \to 0$. In the left corner of the plot, smoothing variance is more than 10^{120} times smaller than filtering variance. These results are different from those of Moore and Teo [200], which assumed the dynamic system to have bounded signal power.

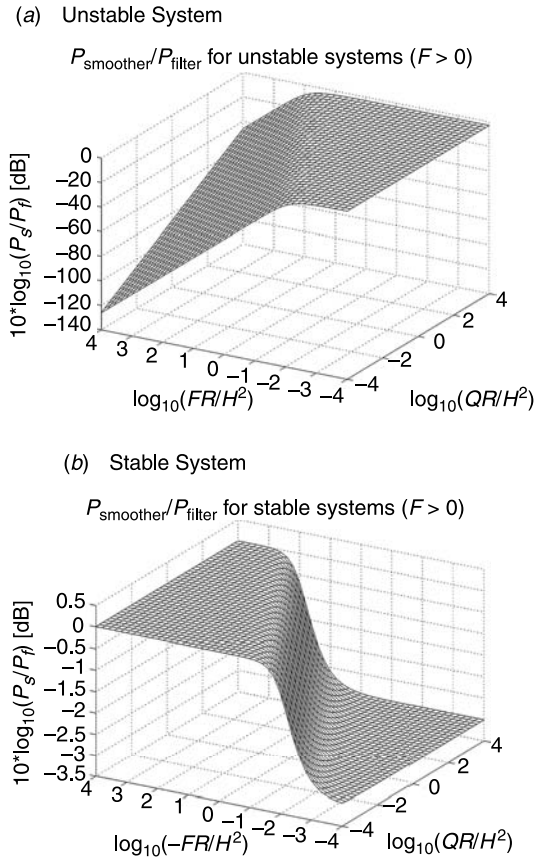


Figure 5.4 Smoothing improvement ratio versus FR/H^2 and QR/H^2.

Random Walk Conditions The smoothing improvement ratio $P_{[s]}/P[f] \rightarrow 1/2$ (−3 dB) as $(FR/H^2) \rightarrow 0$, the random walk model. That is, the improvement of smoothing over filtering for a *random walk* is a factor of 2 decrease in mean-squared estimation uncertainty—or a factor of $\sqrt{2}$ decrease in RMS uncertainty.

Stable Systems For $F < 0$ (Figure 5.4*b*), the smoothing improvement ratio $P_{[s]}/P_{[f]} \rightarrow 1/2$ as $(FR/H^2) \rightarrow 0$, the random walk model. For $F < 0$, $P_{[s]}/P_{[f]} \rightarrow 1/2$ as $(QR/H^2) \rightarrow +\infty$ as well. That is, a factor of 2 improvement in mean-squared smoothing uncertainty over filtering uncertainty is the best one can hope for when $F < 0$. This agrees with the results of Moore and Teo [200].

These plots may provide some insight into how smoothing and filtering performance depend on the parameters of the dynamic system. In more complex models, with matrix-valued parameters F, Q, H, and R, the matrix F can have both positive and negative characteristic values. In that case, the dynamic system model can be

stable in some subspaces of state space and unstable in others, and the dependence of filter covariance $P_{[f]}$ on F can be much more complicated.

5.2.1.3 Finite Interval Case In this case, the measurements $z(t)$ are confined to a finite interval $t_{\text{start}} \le t \le t_{\text{end}}$. This introduces some additional *end effects* on filter and smoother performance. We derive here some formulas for the scalar case ($n = 1$), although the same methods apply to the general case.

This requires the transient solution (as opposed to the steady-state solution) of the Riccati differential equation. For the forward filter, this is

$$P_{[f]}(t) = A_{[f]}(t)B_{[f]}^{-1}(t) \tag{5.7}$$

$$\begin{bmatrix} A_{[f]}(t) \\ B_{[f]}(t) \end{bmatrix} = \exp\left((t - t_{\text{start}}) \begin{bmatrix} F & Q \\ H^2/R & -F \end{bmatrix} \right) \begin{bmatrix} A_{[f]}(t_{\text{start}}) \\ B_{[f]}(t_{\text{start}}) \end{bmatrix} \tag{5.8}$$

for initial conditions $P_{[f]}(t_{\text{start}}) = A_{[f]}(t_{\text{start}})\ B_{[f]}^{-1}(t_{\text{start}})$ at the beginning of the finite interval. Setting $B_{[f]}(t_{\text{start}}) = 0$ and $A_{[f]}(t_{\text{start}}) = 1$ is equivalent to assuming no initial information about the estimate. In that case, the solution will be

$$\tau = \frac{R}{\sqrt{R(F^2R + H^2Q)}} \tag{5.9}$$

$$P_{[f]}(t) = \frac{\sqrt{R(F^2R + H^2Q)}}{H^2} \frac{(e^{(t-t_{\text{start}})/\tau} + e^{-(t-t_{\text{start}})/\tau})}{(e^{(t-t_{\text{start}})/\tau} - e^{-(t-t_{\text{start}})/\tau})} + \frac{FR}{H^2}. \tag{5.10}$$

For the backward filter, this Riccati equation solution is

$$P_{[b]}(t) = A_{[b]}(t)B_{[b]}^{-1}(t) \tag{5.11}$$

$$\begin{bmatrix} A_{[b]}(t) \\ B_{[b]}(t) \end{bmatrix} = \exp\left((t - t_{\text{end}}) \begin{bmatrix} -F & Q \\ H^2/R & F \end{bmatrix} \right) \begin{bmatrix} A_{[b]}(t_{\text{end}}) \\ B_{[b]}(t_{\text{end}}) \end{bmatrix}, \tag{5.12}$$

with the sign of F reversed. As for the forward filter, setting $B_{[b]}\ (t_{\text{end}}) = 0$ and $A_{[b]}\ (t_{\text{end}}) = 1$ is equivalent to assuming no initial information about the estimate at the end of the interval. In that case, the solution will be

$$P_{[b]}(t) = \frac{\sqrt{R(F^2R + H^2Q)}}{H^2} \frac{(e^{(t_{\text{end}}-t)/\tau} + e^{-(t_{\text{end}}-t)/\tau})}{(e^{(t_{\text{end}}-t)/\tau} - e^{-(t_{\text{end}}-t)/\tau})} - \frac{FR}{H^2}. \tag{5.13}$$

The plot in Figure 5.5 was generated using these formulas in the MATLAB m-file `FiniteFIS.m` on the CD. Performance is flat in the midsection because the model is time-invariant. The plot shows the end effects at both ends of the interval, where the smoothing variance kicks up when one of its filters has no information.

You can modify the model parameters in the m-file `FiniteFIS.m` to see how the changes affect smoother and filter performance.

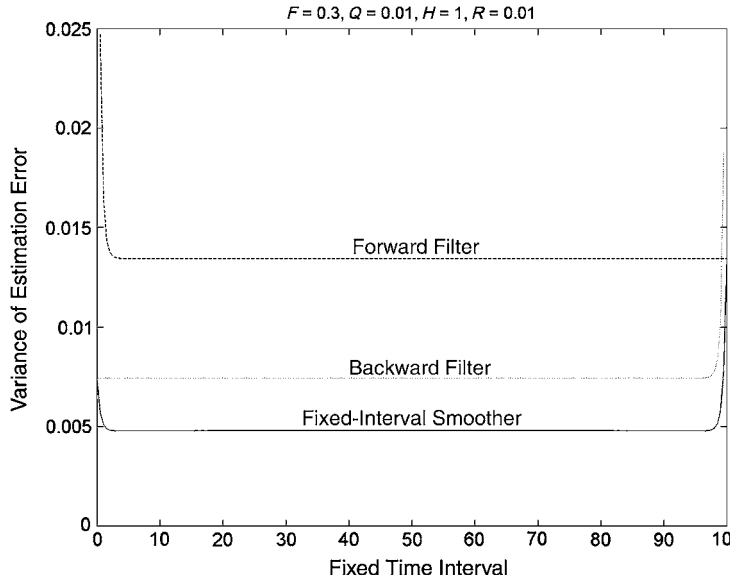


Figure 5.5 Fixed-interval smoothing on finite interval.

5.2.2 Three-Pass Fixed-Interval Smoothing

This is perhaps the oldest of the smoothing methods based on the Kalman filter. It follows the same game plan as the smoother discussed in Section 5.2.1.3, except that it operates in discrete time and it generates the filtered and smoothed estimates as well as their respective covariances of estimation uncertainty.

The general idea is illustrated in Figure 5.6, in which the zigzag paths represent filter outputs of the estimated state vector $\hat{x}$ and its computed covariance matrix P over the discrete time interval from t_1 to t_N. The plot shows the outputs of two such filters, one running forward in time, starting at t_1 on the left and moving to the right, and the other running backward in time, starting at t_N on the right and moving toward the left. At each filter time step (running either forward or backward) the output estimate of each filter represents its best estimate based on all the measurements it has processed up to that time. When the forward and backward filters meet, all the measurements have been used up to produce two estimates. One is based on all the measurements to the left and the other is based on all the measurements to the right. If these two independent estimates can be combined optimally, then the resulting *smoothed estimate* will be based on all the measurements taken throughout the entire fixed interval.

The implementation problem is to do this at every discrete time t_k in the entire fixed interval. It requires three passes through the measurements and data derived therefrom:

1. A complete filter pass in the forward direction (i.e., with measurement time increasing), saving the values of the *a posteriori* estimate $\hat{x}_{[f],k}(+)$ and the

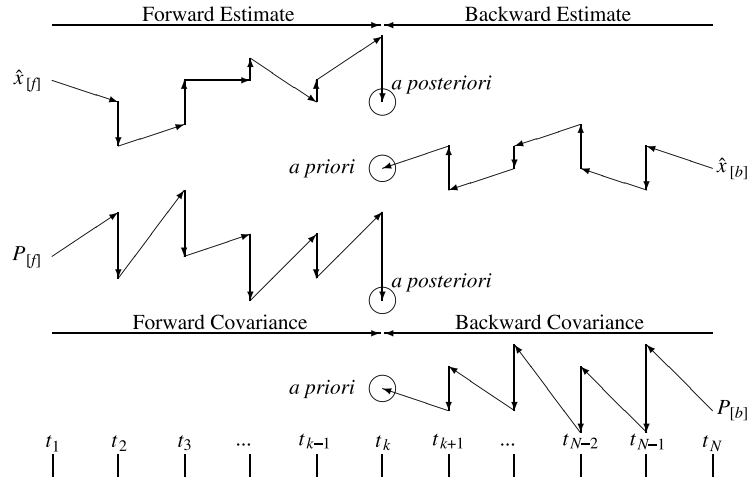


Figure 5.6 Forward and backward filter outputs.

associated covariance of estimation uncertainty $P_{[f],k}(+)$ (the subscript $[f]$ stands for "forward"). The values of the state transition matrices Φ_k can also be saved on this pass.

2. A complete filter pass in the backward direction (time decreasing), saving the *a priori* estimates and associated covariance of estimation uncertainties,

$$\hat{x}_{[b],k-1}(-) = \Phi^{-1}_{[f],k-1}\hat{x}_{[b],k}(+) \text{ (predictor)} \tag{5.14}$$

$$P_{[b],k-1}(-) = \Phi^{-1}_{[f],k-1}P_{[b],k}(+)\Phi^{-\mathrm{T}}_{[f],k-1} + Q_k \tag{5.15}$$

(the subscript [b] stands for "backward"), using the inverses $\Phi^{-1}_{[f],k-1}$ of the state transition matrices from the forward pass (analogous to changing the sign of F in continuous time). The Kalman filter corrector step must also be implemented, although the resulting *a posteriori* estimates and covariances are not used in the third (smoother) pass.

3. A third, smoother pass (which can actually be included with the second pass) combining the forward and backward data to obtain the smoothed estimate

$$\hat{x}_{[s],k}(+) = \hat{x}_{[f],k}(+) + \underbrace{P_{[f],k}(+)[P_{[f],k}(+) + P_{[b],k}(-)]^{-1}}_{\bar{K}}[\hat{x}_{[b],k}(-) - \hat{x}_{[f],k}(+)], \tag{5.16}$$

where $\hat{x}_{s,k}(+)$ is the smoothed estimate.

One can also compute the covariance of smoother uncertainty,

$$P_{[s],k} = \bar{K}P_{[b],k}(-). \tag{5.17}$$

It is not required in the smoother operation, but it is useful for understanding how good the smoothed estimate is supposed to be.

The Smoother Pass Is Not Recursive That is, the variables computed at each time t_k depend only on results saved from the two filter passes, and not on smoother-derived values at adjoining times t_{k-1} or t_{k+1}. The covariance matrix $P_{[s],k}(+)$ of the smoothed estimate can also be computed (without recursion) as an indicator of expected smoother performance, although it is not used in obtaining the smoothed estimate values.

It is important to combine the *a priori* data from one direction with the *a posteriori* data from the other direction, as illustrated in Figure 5.6. This is done to make sure that the forward and backward estimates at each discrete time in the smoothing operation are truly *independent* in that they do not depend on any common measurements.

Optimal Combination of the Two Filter Outputs Equation 5.16 has the form of a Kalman filter observational update (corrector) for combining two statistically independent estimates using their associated covariances of uncertainty, where

$\hat{x}_{[f],k}(+)$ is the effective *a priori* estimate.

$P_{[f],k}(+)$ is its covariance of uncertainty.

$\hat{x}_{[b],k}(-)$ is the effective "measurement" (independent of $\hat{x}^{(+)}_{[f],k}$), with $H = I$ (the identity matrix) its effective measurement sensitivity matrix.

$P_{[b],k}(-)$ is its analogous "measurement noise" covariance (i.e., its covariance of uncertainty).

$\bar{\mathrm{K}} = P_{[f],k}(+)\,[P_{[f],k}(+) + P_{[b],k}(-)]^{-1}$ is the effective Kalman gain.

$\hat{x}_{[s],k}(+)$ is the resulting optimal smoother estimate.

Computational Issues Equation 5.16 requires the inverse of the covariance matrix P from the forward pass, and Equations 5.14 and 5.15 require the inverse of the state transition matrix Φ from the forward pass. The latter (i.e., computing Φ^{-1}) is not a serious problem for time-invariant problems, because Φ would have to be computed and inverted only once. Inversion of the covariance matrix is an issue that has been addressed by developing alternative *information-smoothing* methods based on the information matrix (inverse of P) rather than the covariance matrix P.

The MATLAB m-file `sim3pass.m` on the companion CD generates a trajectory for a scalar linear time-invariant model driven by simulated noise for measurement noise and dynamic disturbance noise, applies a three-pass fixed-interval smoother to the resulting measurement sequence, and plots the resulting estimates along with the true (simulated) values and computed variances of estimation uncertainty. Sample output is shown in Figure 5.7. As an empirical measure of smoothing performance relative to the performance of the two filter, the root mean-squared (RMS) estimation error, computed as the deviation from the true simulated trajectory, is calculated for the forward filter, backward filter, and smoother each time the program is run. However, this smoother is included here primarily to explain the

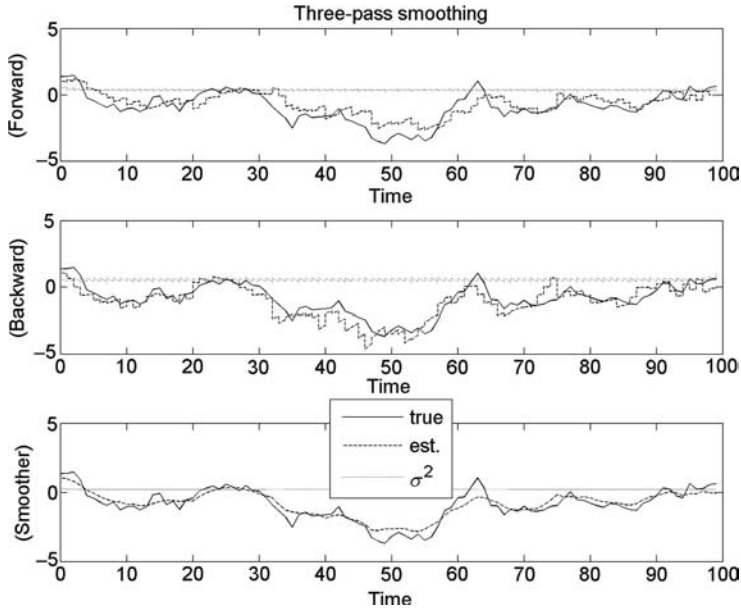


Figure 5.7 Simulated three-pass fixed-interval smoother.

theoretical basis for optimal smoothing, to show that a relatively straightforward implementation method exists and to derive the following example of smoothing performance for very simple models.

5.2.3 Rauch–Tung–Striebel Two-Pass Smoother

This fixed-interval two-pass implementation is the fastest fixed-interval smoother [211], and it has been used quite successfully for decades. The algorithm was publishedby Herbert E. Rauch, F. Tung, and Charlotte T. Striebel in 1965 [228], and the implementation formulas have had some tweaking since then. The first (forward) pass uses a Kalman filter but saves the intermediate results $\hat{x}_k(-)$, $\hat{x}_k(+)$, $P_k(-)$, and $P_k(+)$ at each measurement time t_k. The second pass runs backward in time in a sequence from the time t_N of the last measurement, computing the smoothed state estimate from the intermediate results stored on the forward pass. The smoothed estimate (designated by the subscript $[s]$) is initialized with the value

$$\hat{x}_{[s]N} = \hat{x}_N(+), \tag{5.18}$$

then computed recursively by the formulas

$$\hat{x}_{[s]k} = \hat{x}_k(+) + A_k(\hat{x}_{[s]k+1} - \hat{x}_{k+1}(-)), \tag{5.19}$$

$$A_k = P_k(+)\Phi_k^{\mathrm{T}} P_{k+1}^{-1}(-). \tag{5.20}$$

Computational Complexity Issues Inverting the filter covariance matrices P_k is the major added computational burden. It could also compromise numerical stability when the conditioning of the P_k for inversion is questionable.

Performance The covariance of uncertainty of the smoothed estimate can also be computed on the second pass:

$$P_{[s]k} = P_k(+) + A_k(P_{[s]k+1} - P_{k+1}(-))A_k^{\mathrm{T}}, \tag{5.21}$$

although this is not a necessary part of the smoother implementation. It can be computed if the estimated smoother performance is of some interest.

The MATLAB m-file `RTSvsKF.m`, described in Appendix A, demonstrates (using simulated data) the performance of this smoother relative to that of the Kalman filter.

5.3 FIXED-LAG SMOOTHING

5.3.1 Stability Problems of Early Methods

Rauch published the earliest fixed-lag smoothing method based on the Kalman filter model in 1963 [227]. Kailath and Frost [140] later discovered that the Kalman filter and the Rauch fixed-lag smoother are an *adjoint pair*, and Kelly and Anderson [151] showed this implies that the Rauch fixed-lag smoother is not asymptotically stable. Many of the early fixed-lag smoothing methods were plagued by similar instability problems.

Biswas and Mahalanabis [34] first introduced the idea of augmenting the state vector to recast the smoothing problem as a filtering problem. Biswas and Mahalanabis also proved the stability of their fixed-lag smoother implementation [35] and determined its computational requirements [36].

We present in Section 5.3.3 an augmented state vector implementation from Biswas and Mahalanabis [34], Premier and Vacroux [220], and Moore [198], which has become standard. Prasad and Mahalanabis [219] and Tam and Moore [266] have also developed stable fixed-lag smoothers in continuous time for analog implementations in communications signal processing.

We first introduce a simpler (but less stable) example for the purpose of analyzing how well "generic" fixed-lag smoothing performs relative to filtering, and how relative performance depends on the lag time and other parameters of the application.

5.3.2 Performance Analysis

5.3.2.1 Analytical Model in Continuous Time We can use the scalar Kalman–Bucy filter model of Equations 5.1 and 5.2 from Section 5.2.1 to characterize the relative improvement of fixed-lag smoothing over filtering as a function of lag time and the parameters of the filter model.

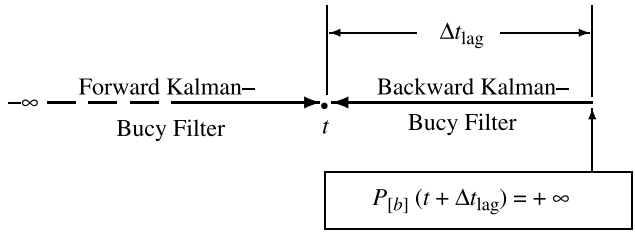


Figure 5.8 Kalman–Bucy fixed-lag smoother model.

How this model is used for the fixed-lag smoother is illustrated in Figure 5.8. At any time t, the effect of the lag time Δt_{lag} on the estimate at time t, given measurements out to time $t + \Delta t_{lag}$, is modeled by an additional filter operating backward from time $t + \Delta t_{lag}$ to time t, starting with no information at all at time $t + \Delta t_{lag}$ (i.e., $P(t + \Delta t_{lag}) = +\infty$). The improvement of fixed-lag smoothing over filtering can then be evaluated as it was in Section 5.2.1, except that this example has as an additional parameter the lag time Δt_{lag}.

5.3.2.2 Forward Kalman–Bucy Filter Performance In Section 5.2.1, the Kalman–Bucy forward filter performance at steady state ($\dot{P}_{[f]} = 0$) was shown to be

$$P_{[f]}(t) = \sqrt{\left(\frac{FR}{H^2}\right)^2 + \left(\frac{QR}{H^2}\right)} + \left(\frac{FR}{H^2}\right), \tag{5.22}$$

which is a formula for filtering performance as a function of the two composite parameters (FR/H^2) and (QR/H^2).

5.3.2.3 Backward Kalman–Bucy Filter Performance For the non-steady-state problem of the backward filter, we can use the linearized Riccati equation solution (originally published by Jacopo Riccati in 1724 [230]) from Section 4.8.3 The solution for the backward filter covariance $P_{[b]}(t)$ at time t, having started at time $t + \Delta t_{lag}$ with initial value $P_{[b]}(t + \Delta t_{lag}) = +\infty$ (i.e., with no initial information, modeled by setting $P(0) = 1$ and the vector component below it $= 0$ in Equation 4.75), is

$$P_{[b]}(t) = \sqrt{\left(\frac{FR}{H^2}\right)^2 + \left(\frac{QR}{H^2}\right)}\frac{e^{\rho} + e^{-\rho}}{e^{\rho} - e^{-\rho}} - \left(\frac{FR}{H^2}\right) \tag{5.23}$$

$$\rho = \left(\frac{H^2 \Delta t_{lag}}{R}\right)\sqrt{\left(\frac{FR}{H^2}\right)^2 + \left(\frac{QR}{H^2}\right)}, \tag{5.24}$$

which introduces a third composite parameter, $(H^2\ \Delta t_{lag}/R)$.

5.3.2.4 Kalman–Bucy Fixed-Lag Smoother Performance The formula for two-filter smoothing (from Section 5.2.1) is

$$P_{[s]}(t) = \frac{P_{[b]}(t)P_{[f]}(t)}{P_{[f]}(t) + P_{[b]}(t)},$$

for the values of $P_{[f]}(t)$ and $P_{[b]}(t)$ derived above as functions of the three composite parameters (FR/H^2), (QR/H^2), and $(H^2\Delta t_{\text{lag}}/R)$.

5.3.2.5 Improvement of Fixed-Lag Smoothing Over Filtering The smoothing improvement ratio over filtering,

$$\frac{P_{[s]}(t)}{P_{[f]}(t)} = \frac{P_{[b]}(t)}{P_{[f]}(t) + P_{[b]}(t)},$$

now depends on three composite parameters: (FR/H^2), (QR/H^2), and $(H^2\,\Delta t_{\text{lag}}/R)$. It cannot be plotted on paper as a function of three parameters, as the infinite-interval smoother performance was plotted as a function of two parameters in Section 5.2.1, but it can still be "plotted" using time as the third dimension. The MATLAB m-file `FLSmovies.m` creates two short video clips of the smoothing improvement ratio $P[s]/P[f]$ as a function of QR/H^2 and FR/H^2 as the lag time parameter $\Delta t_{\text{lag}}\ H^2/R$ varies from frame to frame from 10^{-6} to about 0.06. The created file `FixedLagU.avi` is for $F > 0$. The created file `FixedLagS.avi` is for $F < 0$. These video clips show how fixed-lag smoothing improvement over filtering varies as a function of the three composite parameters (FR/H^2), (QR/H^2), and $(H^2\Delta t_{\text{lag}}/R)$. These files are in Audio Video Interleave format for Windows PCs, and each requires more than a megabyte of storage.

The asymptotic values for fixed-lag smoother performance are

$$\begin{aligned}
&\text{as} \quad \Delta t_{\text{lag}} \to 0, \quad && \frac{P_{[s]}(t)}{P_{[f]}(t)} \to 1 \text{ (no improvement);} \\
&\text{as} \quad \Delta t_{\text{lag}} \to +\infty, \quad && \rho \to +\infty, \\
& && e^{-\rho} \to 0, \\
& && e^{\rho} \to +\infty, \\
& && \frac{e^{\rho} + e^{-\rho}}{e^{\rho} - e^{-\rho}} \to 1, \\
& && \frac{P_{[s]}(t)}{P_{[f]}(t)} \to \frac{\left(\dfrac{QR}{H^2}\right)}{2\left(\dfrac{FR}{H^2}\right)^2 + 2\left(\dfrac{QR}{H^2}\right) + 2\left(\dfrac{FR}{H^2}\right)\sqrt{\left(\dfrac{FR}{H^2}\right)^2 + \left(\dfrac{QR}{H^2}\right)}},
\end{aligned}$$

the improvement ratio of the fixed infinite interval Kalman–Bucy smoother of Section 5.2.1.

5.3.3 Biswas–Mahalanabis Fixed-Lag Smoother (BMFLS)

The earliest implementations for a fixed-lag smoother were mathematically correct but were found to be poorly conditioned for numerical implementation. We demonstrate here the relative numerical stability of an implementation published by Moore [198], based on an approach by Premier and Vacroux [220], which is the *state augmentation* filtering approach published earlier by Biswas and Mahalanabis [34]. It is essentially a Kalman filter using an augmented state vector made up from the successive values of the original system state vector over a discrete time window of fixed width, as illustrated in Figure 5.9. If

$$\Delta t_{\text{lag}} = \ell \Delta t$$

is the time lag, which is ℓ discrete time steps, then the augmented state vector at time t_k is of length $n(\ell+1)$, with $(\ell+1)$ n-dimensional subvectors $x_k, x_{k-1}, x_{k-2}, \ldots, x_{k-\ell}$ —as shown in Figure 5.9.

5.3.3.1 State Vector In order to distinguish between the Kalman filter state vector and the BMFLS state vector, we will use the notation

- $\mathbf{x}_{k,\text{KF}}$ to denote the true value of the state vector of Kalman filter at the kth epoch in discrete time.
- $\hat{\mathbf{x}}_{k,\text{KF}}$ to denote the estimated value of the state vector of Kalman filter at the kth epoch in discrete time.
- $\mathbf{x}_{k,\text{BMFLS}}$, to denote the true value of the state vector of the BMFLS at the kth epoch in discrete time.
- $\hat{\mathbf{x}}_{k,\text{BMFLS}}$ to denote the estimated value of the states vector of the BMFLS at the kth epoch in discrete time.

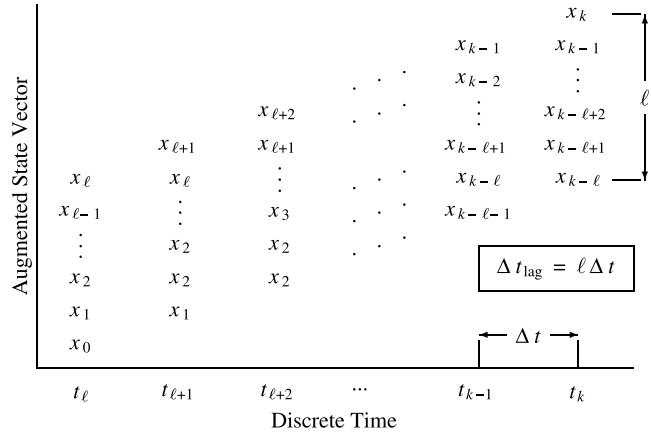


Figure 5.9 Biswas–Mahalanabis augmented state vector.

The BMFLS uses $\ell + 1$ lagged values $\mathbf{x}_{k,\mathrm{KF}}$, $\mathbf{x}_{k-1,\mathrm{KF}}$, $\mathbf{x}_{k-2,\mathrm{KF}}$, $\ldots$, $\mathbf{x}_{k-\ell,\mathrm{KF}}$ of the Kalman filter state vector $\mathbf{x}_{\mathrm{KF}}$, stacked into a single vector of dimension $n\ (\ell+1)\times\ 1$, where n is the number of state variables in the Kalman filter model and ℓ is the fixed number of lag steps in the fixed-lag smoother. That is, the BMFLS state vector is

$$\mathbf{x}_{k,\mathrm{BMFLS}} \stackrel{\mathrm{def}}{=} \begin{bmatrix} \mathbf{x}_{k,\mathrm{KF}} \\ \mathbf{x}_{k-1,\mathrm{KF}} \\ \mathbf{x}_{k-2,\mathrm{KF}} \\ \vdots \\ \mathbf{x}_{k-\ell,\mathrm{KF}} \end{bmatrix}. \tag{5.25}$$

The BMFLS is a Kalman filter using this augmented state vector. If this state vector is estimated using the same sequence of measurements $\{\mathbf{z}_k\}$ as the conventional Kalman filter, then each augmented subvector of the resulting estimate $\hat{\mathbf{x}}_{k,\mathrm{BMFLS}}$ corresponding to $\hat{\mathbf{x}}_{k-\ell,\mathrm{KF}}$ represents the smoothed estimate of $\hat{\mathbf{x}}_{k-\ell}$ using the measurements $\mathbf{z}_j$ for $j \leq k$. That is, the resulting BMFLS estimates the *smoothed* values of

$$\mathbf{x}_{k-\ell,\mathrm{KF}}, \mathbf{x}_{k-\ell+1,\mathrm{KF}}, \mathbf{x}_{k-\ell+2,\mathrm{KF}}, \ldots, \mathbf{x}_{k-1,\mathrm{KF}}$$

plus the *filtered* value $\mathbf{x}_{k,\mathrm{KF}}$, given the measurements

$$\ldots, \mathbf{z}_{k-\ell}, \mathbf{z}_{k-\ell+1}, \mathbf{z}_{k-\ell+2}, \ldots, \mathbf{z}_k.$$

In other words, the estimated state vector of the BMFLS will be equivalent to

$$\hat{\mathbf{x}}_{k,\mathrm{BMFLS}} \equiv \begin{bmatrix} \hat{\mathbf{x}}_{k|k} \\ \hat{\mathbf{x}}_{k-1|k} \\ \hat{\mathbf{x}}_{k-2|k} \\ \vdots \\ \hat{\mathbf{x}}_{k-\ell|k} \end{bmatrix}, \tag{5.26}$$

where $\hat{\mathbf{x}}_{k-\ell|\mathrm{k}}$ is a notation for the smoothed estimate of $\mathbf{x}_{k-\ell}$ given all measurements up to $\mathbf{z}_k$.

5.3.3.2 State Transition Matrix If the Kalman filter model has state transition matrices $\Phi_{k,\mathrm{KF}}$, then the BMFLS state vector has the property that

$$\mathbf{x}_{k+1,\mathrm{BMFLS}} = \begin{bmatrix} \mathbf{x}_{k+1,\mathrm{KF}} \\ \mathbf{x}_{k,\mathrm{KF}} \\ \mathbf{x}_{k-1,\mathrm{KF}} \\ \vdots \\ \mathbf{x}_{k-\ell+2,\mathrm{KF}} \\ \mathbf{x}_{k-\ell+1,\mathrm{KF}} \end{bmatrix}. \tag{5.27}$$

$$= \begin{bmatrix} \Phi_{k,\mathrm{KF}} & 0 & 0 & \cdots & 0 & 0 \\ \mathbf{I} & 0 & 0 & \cdots & 0 & 0 \\ 0 & \mathbf{I} & 0 & \cdots & 0 & 0 \\ 0 & 0 & \mathbf{I} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \mathbf{I} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x}_{k,\mathrm{KF}} \\ \mathbf{x}_{k-1,\mathrm{KF}} \\ \mathbf{x}_{k-2,\mathrm{KF}} \\ \vdots \\ \mathbf{x}_{k-\ell+1,\mathrm{KF}} \\ \mathbf{x}_{k-\ell,\mathrm{KF}} \end{bmatrix}. \tag{5.28}$$

That is, the equivalent state transition matrix for the BMFLS is

$$\boldsymbol{\phi}_{k,\mathrm{BMFLS}} = \begin{bmatrix} \Phi_{k,\mathrm{KF}} & 0 & 0 & \cdots & 0 & 0 \\ \mathbf{I} & 0 & 0 & \cdots & 0 & 0 \\ 0 & \mathbf{I} & 0 & \cdots & 0 & 0 \\ 0 & 0 & \mathbf{I} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \mathbf{I} & 0 \end{bmatrix}. \tag{5.29}$$

5.3.3.3 Process Noise Covariance If $\mathbf{Q}_{k,\mathrm{KF}}$ is the process noise covariance for the Kalman filter model, then

$$\mathbf{Q}_{k,\mathrm{BMFLS}} = \begin{bmatrix} \mathbf{Q}_{k,\mathrm{KF}} & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}. \tag{5.30}$$

5.3.3.4 Measurement Sensitivity Matrix Because the measurement $\mathbf{z}_k$ is sensitive only to the subvector $\hat{\mathbf{x}}_{k,KF}$, the appropriate measurement sensitivity matrix for the BMFLS will be

$$\mathbf{H}_{k,\mathrm{BMFLS}} = [\,\mathbf{H}_{k,\mathrm{KF}} \quad 0 \quad 0 \quad \cdots \quad 0\,], \tag{5.31}$$

where $\mathbf{H}_{k,\mathrm{KF}}$ is the measurement sensitivity matrix from the Kalman filter model.

5.3.3.5 Measurement Noise Covariance Because the measurements for the BMFLS are the same as those for the Kalman filter, the equivalent measurement noise covariance matrix will be

$$\mathbf{R}_{k,\mathrm{BMFLS}} = \mathbf{R}_{k,\mathrm{KF}}, \tag{5.32}$$

the same as for the Kalman filter.

5.3.3.6 *Implementation Equations*

Start-up Transition The BMFLS is actually a filter, but with state augmentation for lagged estimates. During start-up, it must transition from a filter with no augmentation, through augmentation with increasing numbers of lagged estimates, until it reaches the specified fixed number of lags.

At the initial time t_0, this fixed-lag smoother starts with a single initial estimate $\hat{x}_0$ of the system state vector and an initial value P_0 of its covariance of uncertainty.

Let n denote the length of $\hat{x}_0$, so that P_0 will have dimensions $n \times n$, and let ℓ denote the ultimate number of time lags (discrete time steps of delay) desired for fixed-lag smoothing.

State Vector Augmentation As the first $\ell + 1$ measurements $z_1, z_2, z_3, \ldots, z_{\ell+1}$ come in, the smoother incrementally augments its state vector at each time step to build up to $\ell + 1$ time-lagged state vector values in its state vector.

Throughout the smoothing process, the topmost n-component subvector of this augmented state vector will always equal the filtered estimate, and subsequent n-dimensional subvectors are the smoothed estimates with increasing numbers of time lags. The last n components of the smoother state vector will be the smoothed estimate with the current longest lag.

The length of the smoothed estimate state vector increases by n components for the first $\ell + 1$ measurement times t_k, $1 \leq k \leq \ell+1$. At time $t_{\ell+1}$, the specified fixed lag time is reached, the incremental augmentation of the state vector ceases, and the smoother state vector length remains steady at $n(\ell + 1)$ thereafter.

The start-up sequence of state vector augmentation is illustrated in Figure 5.10, where each subvector of length n in the smoother state vector is represented by a separate symbol. Note that the topmost subvector is always the filter estimate of the current state.

If the smoother starts receiving measurements at discrete time t_1, then the length of the smoother state vector will be

$$\text{length } [\hat{x}_{[s]}(t_k)] = \begin{cases} nk, & 1 \leq k \leq \ell + 1 \\ n(\ell + 1), & k \geq \ell + 1, \end{cases} \tag{5.33}$$

and the associated covariance matrix of the smoother state vector uncertainty will grow as $(nk \times nk)$ until $k = \ell + 1$.

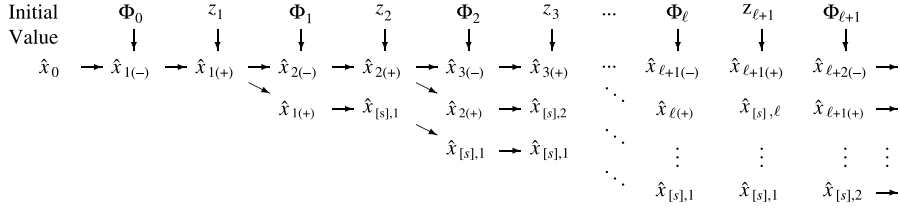


Figure 5.10 Initial state vector transition for BMFLS.

TABLE 5.1 BMFLS Covariance Transition

$P_{[s],k}$	*A Priori* Value	*A Posteriori* Value (dim.)
$P_{[s],1}$	$\mathrm{P}_{[f],1(-)} = \Phi_0 P_0 \Phi_0^T + Q_0$	$P_{[f],1(+)}\ (n \times n)$
$P_{[s],2}$	$\begin{bmatrix} \Phi_1 P_{[f],1(+)} \Phi_1^T + Q_1 & \Phi_1 P_{[f],1(+)} \\ P_{[f],1(+)} \Phi_1^T & P_{[f],1(+)} \end{bmatrix}$	$\begin{bmatrix} P_{[f],2(+)} & P_{[s],1,2(+)} \\ P^T_{[s],1,2(+)} & P_{[s],2,2(+)} \end{bmatrix}$
$P_{[s],3}$	$\begin{bmatrix} \Phi_2 P_{[f],2(+)} \Phi_2^T + Q_2 & \Phi_2 P_{[f],2(+)} & \Phi_2 P_{[s],1,2(+)} \\ P_{[f],2(+)} \Phi_2^T & P_{[f],2(+)} & P_{[s],1,2(+)} \\ P^T_{[s],1,2(+)} \Phi_2^T & P^T_{[s],1,2(+)} & P_{[s],2,2(+)} \end{bmatrix}$	$\vdots$
$\vdots$	$\vdots$	$\vdots$
$P_{[s],\ell}$	$\vdots$	$P_{[s](+)}(t_\ell)(n\ell \times n\ell)$
$P_{[s],\ell+1}$	$\begin{bmatrix} \Phi_{\ell-1} P_{[f],\ell-1(+)} \Phi_{\ell-1}^T & \cdots \\ \vdots & P_{[s](+)}(t_\ell) \end{bmatrix}$	$P_{[s](+)}(t_{\ell+1})$ $(n(\ell+1) \times n(\ell+1))$
$P_{[s],\ell+2}$	$\begin{bmatrix} \Phi_\ell P_{[f],\ell(+)} \Phi_\ell^T & \cdots \\ \cdot & \text{upper-left } (n\ell \times n\ell) \\ \cdot & \text{submatrix of} \\ \cdot & P_{[s](+)}(t_{\ell+1}) \end{bmatrix}$	$P_{[s](+)}(t_{\ell+2})$ $(n(\ell+1) \times n(\ell+1))$

Covariance Matrix Augmentation The corresponding buildup of smoother covariance matrices is shown in Table 5.1—until the size of the matrix stabilizes at $t_{\ell+1}$.

As with the state vector, there are two values of the covariance matrix at each time step t_k: the *a priori* value $P_{[s]}(-)(t_k)$ and the *a posteriori* value $P_{[s]}(+)(t_k)$. A reshuffling of the subblocks of the covariance matrix first occurs with each *a priori* value, with the upper-left subblock of the previous *a posteriori* value becoming the lower-right subblock of the succeeding *a priori* value.

This sort of matrix operation—a shifting of the matrix down along its own diagonal—is known as *displacement*, a concept developed by Prof. Thomas Kailath at Stanford University, who discovered that the rank of the difference between the original and displacement matrices (called the *displacement rank*) characterizes how certain algorithmic operations on matrices can be performed in parallel.

Steady State Given the model parameters as defined above, the implementation equations for the BMFLS are the same as those for the conventional Kalman filter:

$$\mathbf{K}_{k,\mathrm{BMFLS}} = \mathbf{P}_{k,\mathrm{BMFLS}}\, \mathbf{H}^T_{k,\mathrm{BMFLS}}\, (\mathbf{H}_{k,\mathrm{BMFLS}}\, \mathbf{P}_{k,\mathrm{BMFLS}}\, \mathbf{H}^T_{k,\mathrm{BMFLS}} + \mathbf{R}_k)^{-1} \quad \text{Gain matrix}$$

$\hat{\mathbf{x}}_{k,\mathrm{BMFLS}} \leftarrow \hat{\mathbf{x}}_{k,\mathrm{BMFLS}} + \mathbf{K}_{k,\mathrm{BMFLS}} (\mathbf{z}_k - \mathbf{H}_{k,\mathrm{BMFLS}} \hat{\mathbf{x}}_{k,\mathrm{BMFLS}})$ Observational state update

$\mathbf{P}_{k,\mathrm{BMFLS}} \leftarrow \mathbf{P}_{k,\mathrm{BMFLS}} - \mathbf{K}_{k,\mathrm{BMFLS}} \mathbf{H}_{k,\mathrm{BMFLS}} \mathbf{P}_{k,\mathrm{BMFLS}}$ Observational covariance update

$\hat{\mathbf{x}}_{k+1,\mathrm{BMFLS}} \leftarrow \Phi_{k,\mathrm{BMFLS}} \hat{\mathbf{x}}_{k,\mathrm{BMFLS}}$ Temporal state update

$\hat{\mathbf{P}}_{k+1,\mathrm{BMFLS}} \leftarrow \Phi_{k,\mathrm{BMFLS}} \hat{\mathbf{P}}_{k,\mathrm{BMFLS}} \Phi^{\mathrm{T}}_{k,\mathrm{BMFLS}} + \mathbf{Q}_{k,\mathrm{BMFLS}}$ Temporal covariance update,

where $\mathbf{P}_{k,\mathrm{BMFLS}}$ is the covariance matrix of smoother estimation uncertainty. In the following application to exponentially correlated noise, an initial value for $\mathbf{P}_{k,\mathrm{BMFLS}}$ is determined from the steady-state value of the Riccati equation without measurements.

MATLAB Implementation The MATLAB function `BMFLS.m` on the accompanying CD implements the complete BMFLS, from the first measurement, through the transition to steady-state state augmentation, and after that.

Example 5.1 (MATLAB Example Using `BMFLS.m`) Figure 5.11 is a plot of a sample application of `BMFLS.m` using simulated data. The model in this case is for a sinusoidal signal being demodulated and sampled in-phase and in quadrature with phase slip rate ω. The problem is time-invariant. The state vector and the measurement vector both have two components, and the signal model is

$$\Phi = \exp(-\Delta t/\tau) \begin{bmatrix} \cos(\omega \Delta t) & \sin(\omega \Delta t) \\ -\sin(\omega \Delta t) & \cos(\omega \Delta t) \end{bmatrix} \tag{5.34}$$

$$x_k = \Phi x_{k-1} + w_{k-1} \tag{5.35}$$

$$z_k = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} x_k + v_k, \tag{5.36}$$

with other parameters as specified in Problem 5.8. The plots include both phase components, as well as the estimates and associated RMS uncertainties.

Note that RMS smoothing uncertainty for this example (which is stable) is about 20% less than filtering uncertainty.

Example 5.2 (Algebraic Riccati Equation Solution) The algebraic Riccati equation for the BMFLS with a scalar state vector has a relatively regular structure, from which it is possible to obtain a closed-form solution.

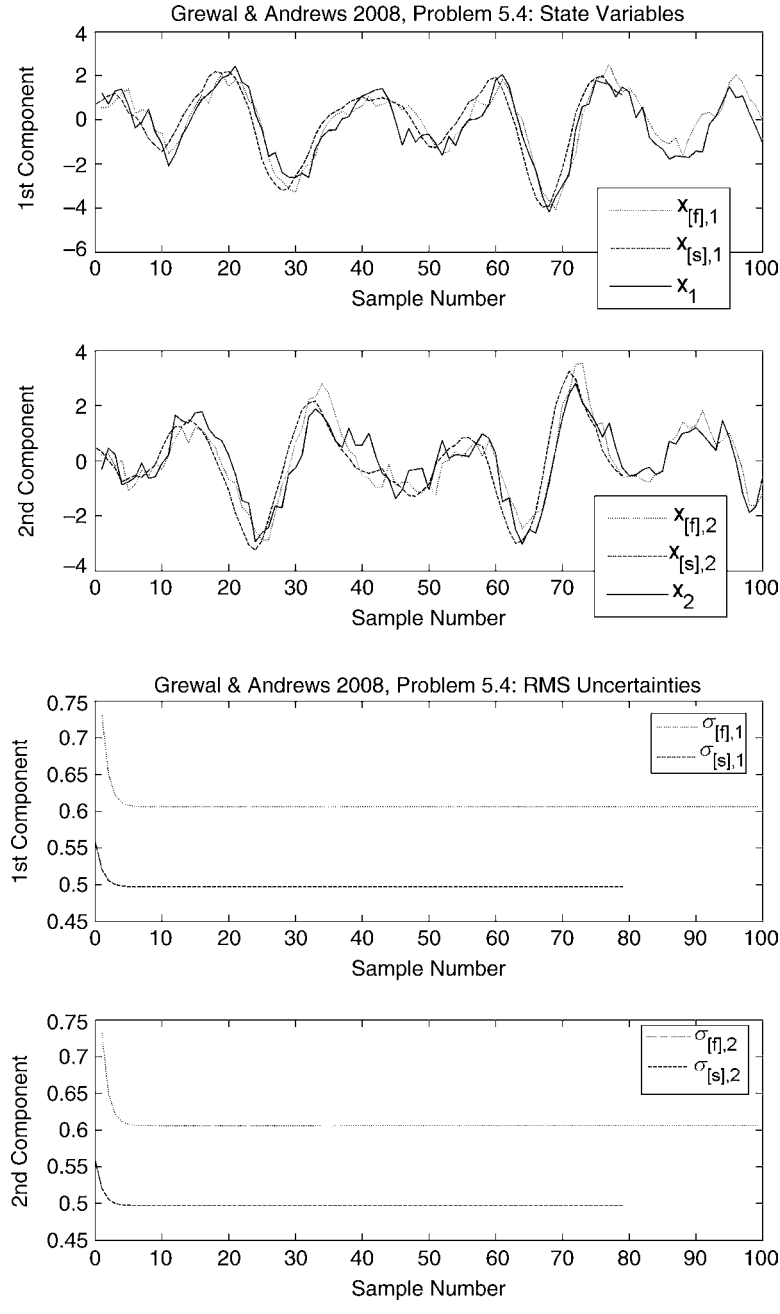


Figure 5.11 Example application of `BMFLS.m`.

The smoother state transition matrix in this case has the form

$$\Phi_{[s]} = \begin{bmatrix} \phi & 0 & 0 & \cdots & 0 & 0 \\ 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \tag{5.37}$$

$$\phi = \exp(\Delta t/\tau), \tag{5.38}$$

where ϕ is a scalar.

Let the *a posteriori* smoother covariance matrix be

$$P_{[s],k-1}(+) = \begin{bmatrix} p_{1,1} & p_{1,2} & p_{1,3} & \cdots & p_{1,\ell} & p_{1,\ell+1} \\ p_{1,2} & p_{2,2} & p_{2,3} & \cdots & p_{2,\ell} & p_{2,\ell+1} \\ p_{1,3} & p_{2,3} & p_{3,3} & \cdots & p_{3,\ell} & p_{3,\ell+1} \\ p_{1,4} & p_{2,4} & p_{3,4} & \cdots & p_{3,\ell} & p_{3,\ell+1} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ p_{1,\ell} & p_{2,\ell} & p_{3,\ell} & \cdots & p_{\ell,\ell} & p_{\ell,\ell+1} \\ p_{1,\ell+1} & p_{2,\ell+1} & p_{3,\ell+1} & \cdots & p_{\ell,\ell+1} & p_{\ell+1,\ell+1} \end{bmatrix}, \tag{5.39}$$

so that the matrix product is

$$\Phi_{[s]}P_{[s],k-1}(+) = \begin{bmatrix} \phi p_{1,1} & \phi p_{1,2} & \phi p_{1,3} & \cdots & \phi p_{1,\ell} & \phi p_{1,\ell+1} \\ p_{1,1} & p_{1,2} & p_{1,3} & \cdots & p_{1,\ell} & p_{1,\ell+1} \\ p_{1,2} & p_{2,2} & p_{2,3} & \cdots & p_{2,\ell} & p_{2,\ell+1} \\ p_{1,3} & p_{2,3} & p_{3,3} & \cdots & p_{3,\ell} & p_{3,\ell+1} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ p_{1,\ell-1} & p_{2,\ell-1} & p_{3,\ell-1} & \cdots & p_{\ell-1,\ell} & p_{\ell-1,\ell+1} \\ p_{1,\ell} & p_{2,\ell} & p_{3,\ell} & \cdots & p_{\ell,\ell} & p_{\ell,\ell+1} \end{bmatrix} \tag{5.40}$$

and the *a priori* smoother covariance matrix is

$$P_{[s],k}(-) = \Phi_{[s]}P_{[s],k-1}(+)\Phi_{[s]}^{\mathrm{T}} + Q_{[s]} \tag{5.41}$$

$$= \begin{bmatrix} \phi^2 p_{1,1} + q & \phi p_{1,1} & \phi p_{1,2} & \cdots & \phi p_{1,\ell-1} & \phi p_{1,\ell} \\ \phi p_{1,1} & p_{1,1} & p_{1,2} & \cdots & p_{1,\ell-1} & p_{1,\ell} \\ \phi p_{1,2} & p_{1,2} & p_{2,2} & \cdots & p_{2,\ell-1} & p_{2,\ell} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \phi p_{1,\ell-1} & p_{1,\ell-1} & p_{2,\ell-1} & \cdots & p_{\ell-1,\ell-1} & p_{\ell-1,\ell} \\ \phi p_{1,\ell} & p_{1,\ell} & p_{2,\ell} & \cdots & p_{\ell-1,\ell} & p_{\ell,\ell} \end{bmatrix}. \tag{5.42}$$

The measurement sensitivity matrix is

$$H_{[s]} = [\,1 \quad 0 \quad 0 \quad \cdots \quad 0\,], \tag{5.43}$$

and the matrix product

$$P_{[s],k}(-)H_{[s]}^{\mathrm{T}} = \begin{bmatrix} \phi^2 p_{1,1} + q \\ \phi p_{1,1} \\ \phi p_{1,2} \\ \vdots \\ \phi p_{1,\ell-1} \\ \phi p_{1,\ell} \end{bmatrix} \tag{5.44}$$

and the innovations variance (a scalar)

$$H_{[s]}P_{[s],k}(-)H_{[s]}^{\mathrm{T}} + R_{[s]} = \phi^2 p_{1,1} + q + r \tag{5.45}$$

so that the matrix product

$$P_{[s],k}(-)H_{[s]}^{\mathrm{T}}[H_{[s]}P_{[s],k}(-)H_{[s]}^{\mathrm{T}} + R]^{-1}H_{[s]}P_{[s],k}(-) = \frac{1}{p_{1,1}\phi^2 + q + r}$$
$$\begin{bmatrix} (p_{1,1}\phi^2 + q)^2 & (p_{1,1}\phi^2 + q)p_{1,1}\phi & (p_{1,1}\phi^2 + q)p_{1,2}\phi & \cdots & (p_{1,1}\phi^2 + q)p_{1,\ell}\phi \\ (p_{1,1}\phi^2 + q)p_{1,1}\phi & {p_{1,1}}^2\phi^2 & p_{1,1}\phi^2 p_{1,2} & \cdots & p_{1,1}\phi^2 p_{1,\ell} \\ (p_{1,1}\phi^2 + q)p_{1,2}\phi & p_{1,1}\phi^2 p_{1,2} & {p_{1,2}}^2\phi^2 & \cdots & p_{1,2}\phi^2 p_{1,\ell} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ (p_{1,1}\phi^2 + q)p_{1,\ell}\phi & p_{1,1}\phi^2 p_{1,\ell} & p_{1,2}\phi^2 p_{1,\ell} & \cdots & {p_{1,\ell}}^2\phi^2 \end{bmatrix} \tag{5.46}$$

and the *a posteriori* smoother covariance

$$P_{[s],k}(+) = P_{[s],k}(-) - P_{[s],k}(-)H_{[s]}^{\mathrm{T}}[H_{[s]}P_{[s],k}(-)H_{[s]}^{\mathrm{T}} + R]^{-1}H_{[s]}P_{[s],k}(-) \tag{5.47}$$
$$= \begin{bmatrix} \phi^2 p_{1,1} + q & \phi p_{1,1} & \phi p_{1,2} & \cdots & \phi p_{1,\ell} \\ \phi p_{1,1} & p_{1,1} & p_{1,2} & \cdots & p_{1,\ell} \\ \phi p_{1,2} & p_{1,2} & p_{2,2} & \cdots & p_{2,\ell} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \phi p_{1,\ell} & p_{1,\ell} & p_{2,\ell} & \cdots & p_{\ell,\ell} \end{bmatrix} - \frac{1}{p_{1,1}\phi^2 + q + r}$$
$$\begin{bmatrix} (p_{1,1}\phi^2 + q)^2 & (p_{1,1}\phi^2 + q)p_{1,1}\phi & (p_{1,1}\phi^2 + q_{1,2}\phi & \cdots & (p_{1,1}\phi^2 + q)p_{1,\ell}\phi \\ (p_{1,1}\phi^2 + q)p_{1,1}\phi & {p_{1,1}}^2\phi^2 & p_{1,1}\phi^2 p_{1,2} & \cdots & p_{1,1}\phi^2 p_{1,\ell} \\ (p_{1,1}\phi^2 + q)p_{1,2}\phi & p_{1,1}\phi^2 p_{1,2} & {p_{1,2}}^2\phi^2 & \cdots & p_{1,2}\phi^2 p_{1,\ell} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ (p_{1,1}\phi^2 + q)p_{1,\ell}\phi & p_{1,1}\phi^2 p_{1,\ell} & p_{1,2}\phi^2 p_{1,\ell} & \cdots & {p_{1,\ell}}^2\phi^2 \end{bmatrix}.$$

At steady-state, the *a posteriori* smoother covariance matrix of Equation 5.47 will equal the *a posteriori* smoother covariance matrix in Equation 5.39. By equating the matrix elements term by term, one gets the $\ell(\ell+1)/2$ equations:

$$p_{1,1} = p_{1,1}\phi^2 + q - \frac{(p_{1,1}\phi^2 + q)^2}{p_{1,1}\phi^2 + q + r} \tag{5.48}$$

$$p_{1,i+1} = p_{1,i}\phi - \frac{(p_{1,1}\phi^2 + q)p_{1,i}\phi}{p_{1,1}\phi^2 + q + r} \tag{5.49}$$

$$p_{i+1,i+1} = p_{i,i} - \frac{p_{1,i}^2\phi^2}{p_{1,1}\phi^2 + q + r} \tag{5.50}$$

$$p_{j+1,j+k+1} = p_{j,j+k} - \frac{p_{1,j}\phi^2 p_{1,j+k}}{p_{1,1}\phi^2 + q + r} \tag{5.51}$$

for $1 \le i \le \ell$, $1 \le j \le \ell - 1$ and $1 \le k \le \ell - j$.

These can all be solved recursively, starting with the variance of steady-state filtering uncertainty,

$$p_{1,1} = \frac{\phi^2 r - q - r + \sqrt{r^2(1-\phi^2)^2 + 2rq(\phi^2+1) + q^2}}{\phi^2}, \tag{5.52}$$

then with the cross-covariances of filter error with the smoother estimates,

$$p_{1,i+1} = \frac{p_{1,i}\phi r}{p_{1,1}\phi^2 + q + r}, \tag{5.53}$$

for $1 \le i \le \ell$. Next, the variances of smoother stage estimates,

$$p_{i+1,i+1} = \frac{(p_{i,i}p_{1,1} - p_{1,i}^2)\phi^2 + p_{i,i}(q+r)}{p_{1,1}\phi^2 + q + r}, \tag{5.54}$$

and finally the covariances between different smoother stage errors,

$$p_{j+1,j+k+1} = \frac{(p_{j,j+k}p_{1,1} - p_{1,j}p_{1,j+k})\phi^2 + p_{j,j+k}(q+r)}{p_{1,1}\phi^2 + q + r} \tag{5.55}$$

for $1 \le j \le \ell - 1$ and $1 \le k \le \ell - j$.

The ratio of ℓ-stage smoother variance to filter variance is then

$$\frac{\sigma_{[s]}^2}{\sigma_{[f]}^2} = \frac{p_{\ell+1,\ell+1}}{p_{1,1}}. \tag{5.56}$$

If the scalar $|\phi| < 1$, the dynamic system model is stable and models an exponentially correlated random process with a finite steady-state mean-squared signal. Then the Riccati equation solution can be computed as a function of the SNR and the ratio

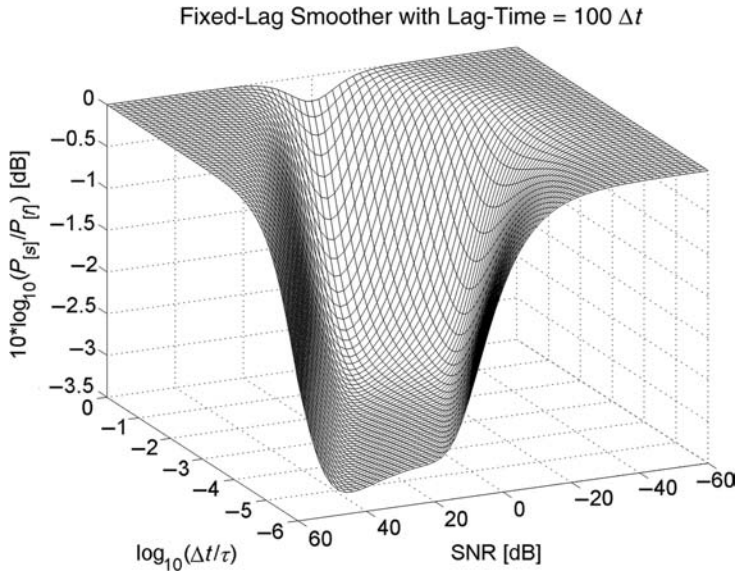


Figure 5.12 Improvement of smoothing over filtering versus SNR and δ/τ.

of discrete time step to correlation time. An example plot is shown in Figure 5.12, which plots the improvement of fixed-lag smoothing over filtering as a function of SNR and $\Delta\ t/\tau$, where τ is the process correlation time. Note that the best improvement is a factor of 2 (-3 dB), which is as good as it gets for stable systems.

5.4 FIXED-POINT SMOOTHING

5.4.1 Performance Analysis

Performance Models Performance models are just that. They model estimator performance. They are based on an estimator model but do not include the estimator itself. They only include the equations which govern the propagation of the covariance of estimation uncertainty.

Scalar Continuous Linear Time-Invariant Model The following analytical performance model is for the scalar linear time-invariant case in continuous time. The stochastic system is modeled in continuous time because that problem is generally easier to manipulate and solve mathematically. The resulting formulas define how fixed-point smoothing performs as a function of time t, the scalar parameters F, Q, H, and R, and the time t_{fixed} at which the smoothed estimated is required. Performance is defined by the variance of estimation uncertainty for the fixed-point smoother.

Comparisons Between Smoothing and Filtering Because there is no comparable filtering method that estimates the system state at a fixed time, we cannot compare fixed-lag smoothing performance with filter performance. Also, because we have added yet another model parameter (t_{fixed}), we can no longer reduce the number of independent parameters and plot smoother performance as a function of the model parameters, as was done for fixed-interval smoothing and fixed-lag smoothing (by resorting to videos).

MATLAB Implementations This same model was used in generating the figures of RMS fixed-point smoother estimation uncertainty versus time. The enclosed MATLAB m-file `FPSperformance.m` implements the formulas used in generating the figures. You can modify this script to vary the model parameters and observe how that changes smoother performance.

5.4.1.1 End-to-End Model for Fixed-Point Smoothing The all-inclusive end-to-end fixed-point smoother model is made up of three different models, as illustrated in Figure 5.13. This shows a *forward Kalman–Bucy filter*, a *Kalman–Bucy predictor* from times before the designated fixed point in time t_{fixed}, and a *backward Kalman–Bucy filter* for times $t > t_{\text{fixed}}$. Different submodels are used over different time intervals:

1. For $t_{\text{start}} \leq t < t_{\text{fixed}}$, the variance of estimation uncertainty from the forward filter at time t is shown as $P_{[f]}(t)$. This is determined by the forward Kalman–Bucy filter model. The result must be projected ahead to the fixed time t_{fixed} by the prediction model, as $P_{[s]}(t_{\text{fixed}}) = P_{[p]}(t_{\text{fixed}})$, the variance of fixed-point smoothing uncertainty during that time period. The prediction uncertainty variance $P_{[p]}(t_{\text{fixed}})$ is a function of the filter uncertainty variance $P_{[f]}(t)$, as determined by the predictor model. The transformation of the filter uncertainty variance $P_{[f]}(t)$ at time t to the prediction uncertainty variance $P_{[p]}(t_{\text{fixed}})$ at time t_{fixed} uses the linearized Riccati equation models of Section 4.8.3 with $H = 0$ (i.e., no measurements).
2. When $t = t_{\text{fixed}}$, the variance of fixed-point smoothing uncertainty will be $P_{[s]}(t_{\text{fixed}}) = P_{[f]}(t_{\text{fixed}})$, the variance of the forward filter estimation uncertainty.

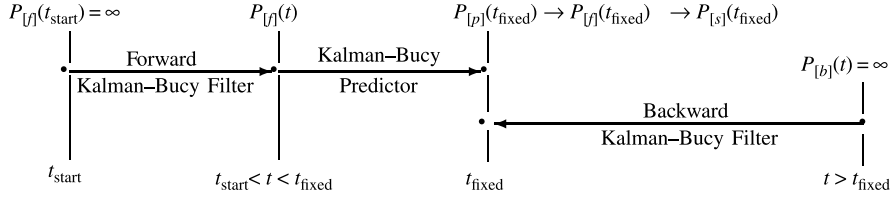


Figure 5.13 Kalman–Bucy fixed-point smoother model.

3. When $t > t_{\text{fixed}}$, the variance of fixed-point smoothing uncertainty will be

$$P_{[s]}(t_{\text{fixed}}) = \frac{P_{[f]}(t_{\text{fixed}})P_{[b]}(t_{\text{fixed}})}{P_{[f]}(t_{\text{fixed}}) + P_{[b]}(t_{\text{fixed}})},$$

a combination of the forward ($P_{[f]}$) and backward ($P_{[b]}$) filter variances.

These submodels are further described and derived in Sections 5.4.1.2 through 5.4.1.5.

Simpler Models for Special Cases Not all fixed-point smoothing problems require all the parts of this model:

1. If $t_{\text{fixed}} = t_{\text{start}}$, the initial time, then the forward filter and predictor are not required. This would be the case, for example, for estimating initial conditions using later measurements—as in the aforementioned problem of determining initial inertial navigation alignment errors.
2. If $t_{\text{fixed}} \geq t$ for the entire mission, then the backward filter is not required. This would be the case for a missile intercept, for example, in which the predicted positions of the interceptor and target at $t_{\text{fixed}} =$ the predicted time of impact are the state variables of primary interest for intercept guidance.

5.4.1.2 Forward Filter Submodel The subscript $[f]$ is used for parameters and variables of the forward Kalman–Bucy filter model. If there is no information about the state variable at the beginning, then the linearized Riccati equation for the Kalman–Bucy filter has the form

$$\begin{aligned}
P_{[f]}(t) &= A_{[f]}(t)/B_{[f]}(t) \\
\begin{bmatrix} A_{[f]}(t) \\ B_{[f]}(t) \end{bmatrix} &= \exp\left(\begin{bmatrix} F & Q \\ H^2/R & -F \end{bmatrix} (t - t_{\text{start}}) \right) \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\
P_{[f]}(t) &= \frac{E^{(+)}_{[f]}\rho_{[f]} - RFE^{(-)}_{[f]} + RFE^{(+)}_{[f]} + E^{(-)}_{[f]}\rho_{[f]}}{H^2(-E^{(-)}_{[f]} + E^{(+)}_{[f]})} \\
\rho_{[f]} &= \sqrt{R(F^2R + QH^2)} \\
E^{(-)}_{[f]} &= e^{-\rho[f](t-t_{\text{start}})/R} \\
E^{(+)}_{[f]} &= e^{\rho[f](t-t_{\text{start}})/R}.
\end{aligned}$$

As part of the model design assumptions, as $t \to t_{\text{start}}$, $P_{[f]}(t) \to +\infty$. Therefore, for this model, RMS estimation uncertainty for the forward filter is valid only for evaluation times $t_{\text{start}} < t \leq t_{\text{fixed}}$.

5.4.1.3 Prediction Submodel The subscript $[p]$ is used for parameters and variables of the Kalman–Bucy predictor model. The predictor carries forward the filtered variance $P_{[f]}(t)$ to time $t_{\text{fixed}} > t$, assuming no measurements. The model in this case becomes

$$
\begin{aligned}
P_{[p]}(t_{\text{fixed}}) &= A_{[p]}(t_{\text{fixed}})/B_{[p]}(t_{\text{fixed}}) \\
\begin{bmatrix} A_{[p]}(t_{\text{fixed}}) \\ B_{[p]}(t_{\text{fixed}}) \end{bmatrix} &= \exp\left(\begin{bmatrix} F & Q \\ 0 & -F \end{bmatrix} (t_{\text{fixed}} - t) \right) \begin{bmatrix} P_{[f]}(t) \\ 1 \end{bmatrix} \\
P_{[p]}(t_{\text{fixed}}) &= e^{2\,(t_{\text{fixed}}-t)F} P_{[f]}(t) + \frac{Q(e^{2\,(t_{\text{fixed}}-t)F} - 1)}{2\,F} \\
&= P_{[f]}(t) + Q(t_{\text{fixed}} - t) \quad \text{if } F = 0 \\
&= P_{[f]}(t_{\text{fixed}}) \quad \text{if } t = t_{\text{fixed}}.
\end{aligned}
$$

5.4.1.4 Backward Filter Submodel Fixed-point smoothing is not implemented using a backward Kalman filter, but the performance of the fixed-point smoother can be characterized by such a model.

The subscript $[b]$ is used for parameters and variables of the backward Kalman–Bucy filter model. This works backward from $t > t_{\text{fixed}}$ to t_{fixed}, starting with no state variable information ($P_{[b]}(t) = \infty$ at t). Reversing the direction of time changes the Riccati differential equation by reversing the sign of F, so that the linearized Riccati equation model becomes

$$
\begin{aligned}
P_{[b]}(t_{\text{fixed}}) &= A_{[b]}(t_{\text{fixed}})/B_{[b]}(t_{\text{fixed}}) \\
\begin{bmatrix} A_{[b]}(t_{\text{fixed}}) \\ B_{[b]}(t_{\text{fixed}}) \end{bmatrix} &= \exp\left(\begin{bmatrix} -F & Q \\ H^2/R & F \end{bmatrix} (t - t_{\text{fixed}}) \right) \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\
P_{[b]}(t_{\text{fixed}}) &= \frac{E^{(+)}_{[b]}\,\rho_{[b]} + RFE^{(-)}_{[b]} - RFE^{(+)}_{[b]} + E^{(-)}_{[b]}\,\rho_{[b]}}{H^2(-E^{(-)}_{[b]} + E^{(+)}_{[b]})} \\
\rho_{[b]} &= \sqrt{R(F^2R + QH^2)} \\
E^{(-)}_{[b]} &= e^{-\rho_{[b]}\,(t-t_{\text{fixed}})/R} \\
E^{(+)}_{[b]} &= e^{\rho_{[b]}\,(t-t_{\text{fixed}})/R}.
\end{aligned}
$$

5.4.1.5 Fixed-Point Smoother Model The resulting fixed-point smoother model uses the forward-filter, predictor, and backward-filter models, depending on where t is relative to t_{fixed}:

$$
P_{[s]}(t_{\text{fixed}}) = \begin{cases} P_{[p]}(t_{\text{fixed}}), & t < t_{\text{fixed}} \\ P_{[f]}(t_{\text{fixed}}), & t = t_{\text{fixed}}, \\ \dfrac{P_{[f]}(t_{\text{fixed}})P_{[b]}(t_{\text{fixed}})}{P_{[f]}(t_{\text{fixed}}) + P_{[b]}(t_{\text{fixed}})}, & t > t_{\text{fixed}}, \end{cases}
$$

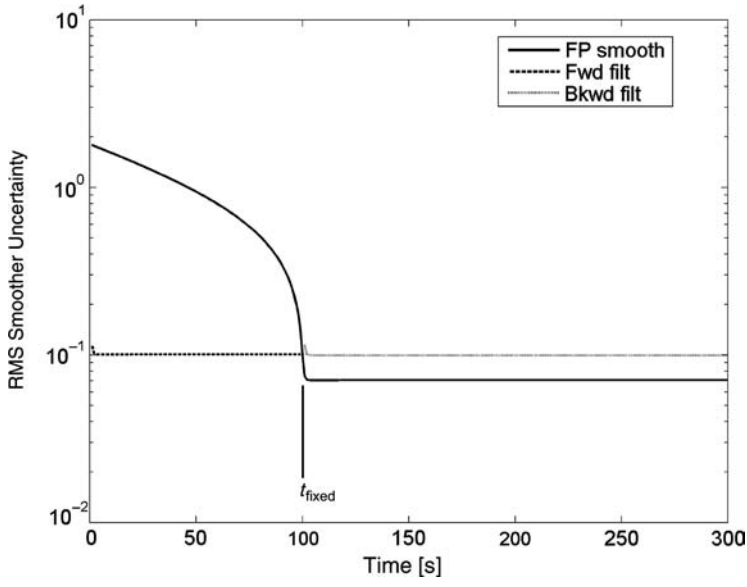


Figure 5.14 Sample performance of the fixed-point smoother with $t_{\text{fixed}} = 100$.

where the subscript $[s]$ on $P_{[s]}$ indicates the variance of estimation uncertainty of the fixed-point smoother.

5.4.1.6 Fixed-Point Smoothing Performance Figure 5.14 is a plot of modeled RMS fixed-point smoother uncertainty for a stochastic process model with

$F = 0.01$ (dynamic coefficient)
$Q = 0.01$ (variance of dynamic process noise)
$H = 1$ (measurement sensitivity)
$R = 0.01$ (variance of measurement noise)
$t_{\text{start}} = 0$ (starting time)
$t_{\text{fixed}} = 100$ (fixed time at which the system state is to be estimated)
$t_{\text{end}} = 300$ (ending time)

The plot also shows the RMS uncertainties in the forward filter estimate and the backward filter estimate. Notice how slowly the RMS uncertainty in the smoothed estimate falls at first, then much faster as the current time approaches the fixed time. For this particular set of parameters, it is the predictor which limits the RMS uncertainty of the smoother, which falls by another $1/\sqrt{2}$ as time passes t_{fixed} and the backward filter kicks in. The plot also indicates that, for these particular parameter values, the backward filter does not continue to reduce smoother uncertainty by much beyond a few seconds past t_{fixed}.

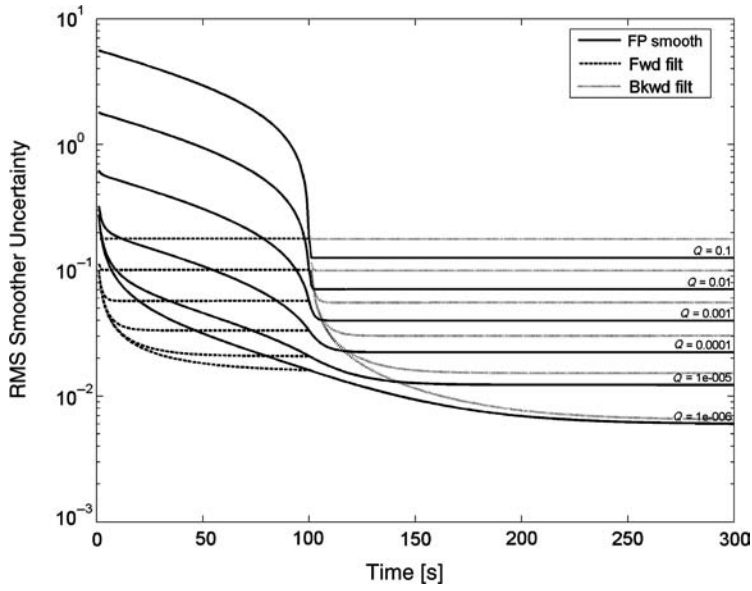


Figure 5.15 Fixed-point smoother performance with varying Q.

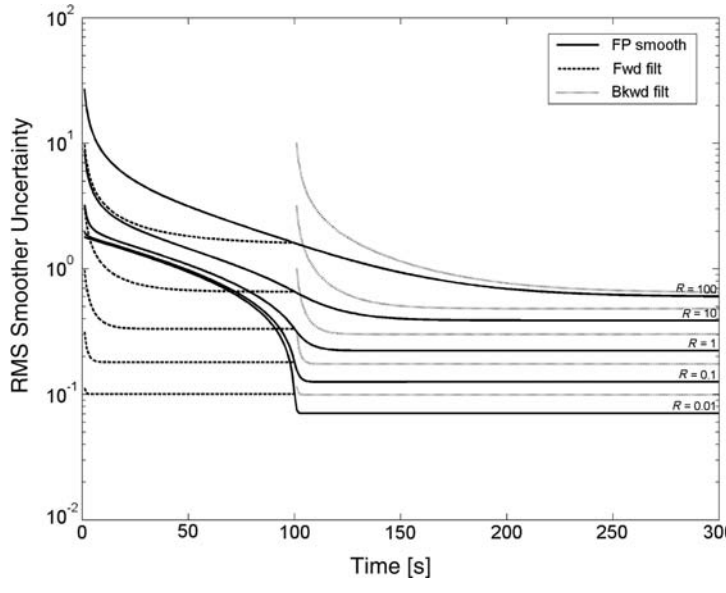


Figure 5.16 Fixed-point smoother performance with varying R.

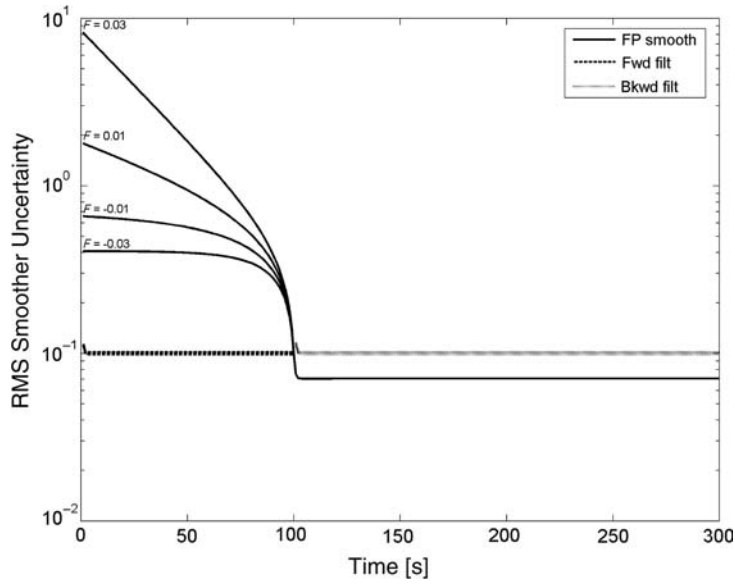


Figure 5.17 Fixed-point smoother performance with varying F.

This plot was generated by the MATLAB m-file `FPSperformance.m`. You can vary these parameter values to see how they influence performance.

The effects of varying the model parameters are illustrated in Figures 5.15 (varying Q), 5.16 (varying R), and 5.17 (varying F), which are multiplots of the results with different parameter values.

5.4.2 Discrete-Time Fixed-Point Smoother

This type of smoother includes a Kalman filter to estimate the state at the current time t_k using the measurements up to time t_k, then adds the following equations to obtain a smoothed estimate of the state at a fixed time $t_i < t_k$:

$$\hat{x}_{[s]i|k} = \hat{x}_{[s]i|k-1} + B_k \overline{K}_k (z_k - H\hat{x}k(-)), \tag{5.57}$$

$$B_k = B_{k-1} P_{k-1}(+) \Phi_{k-1}^{\mathrm{T}} P_k^{-1}(-), \tag{5.58}$$

where the subscript notation $[s]i|k$ refers to the smoothed estimate of the state at time t_i, given the measurements up to time t_k. (A derivation and application of this technique to the analysis of inertial navigation system test data may be found in [104].) The values of $\hat{x}(-)$, $\bar{K}_k$, z_k, H_kP, and P are computed in the Kalman filter and the initial value $B_i = I$, the identity matrix. The covariance of uncertainty of the

smoothed estimate can also be computed by the formula

$$P_{[s]i|k} = P_{[s]i|k-1} + B_k(P_k(+) - P_k(-))B_k^{\mathrm{T}}, \tag{5.59}$$

although this is not a necessary part of the smoother implementation.

5.5 SUMMARY

5.5.1 Smoothing

Modern optimal smoothing methods have been derived from the same types of models used in Kalman filtering. As a rule, the smoothed estimate has smaller mean-squared uncertainty than the corresponding filter estimate, but it requires measurements made *after* the time that the estimate is valid. Smoothing implementations are called *smoothers*. Several implementation algorithms have been derived, some with lower computational requirements or better numerical stability than others. Some of the more stable methods have been presented and implemented in MATLAB m-files.

5.5.1.1 Types of Smoothers Different types of smoothers have been developed for different types of applications. The most common types are:

Fixed-interval smoothers, which obtain the optimal estimate each time a measurement is sampled, using all the measurements sampled in the entire interval. They are most commonly used in postprocessing.

Fixed-lag smoothers, which typically run in real time but generate an estimate in delayed time. That is, when a measurement is sampled at time t, it is used to generate an estimate of the system state vector at time $t - \Delta t_{\mathrm{lag}}$. Some of the first fixed-lag smoother implementations were numerically unstable, a problem which has been solved by more recent implementation methods.

Fixed-point smoothers, which generate estimates of the system state vector at a fixed time t_{fixed}, using measurements made at times before and after t_{fixed}.

5.5.1.2 Implementations Smoothers can be implemented in discrete time (as algorithms) or in continuous time (as analog circuits) If the input signal bandwidth is too high for sampling, or computational requirements preclude digital implementation, then Kalman–Bucy models in continuous time can be used for designing the implementation circuitry.

In either case, there can be alternative realizations of the same smoothing function with different stability characteristics and different levels of complexity. Those implementation methods with less sensitivity to model parameter values and roundoff errors—or to analog component variability and noise—are preferred.

5.5.2 Important Points to Remember

The metric used for the improvement of smoothing over filtering is the ratio of mean-squared estimation uncertainties.

- The relative improvement of smoothing over filtering depends on the values of the parameters of the stochastic system model (Q and F or Φ) and the sensor model (H and R).
- For stable dynamic system models, the improvement of smoothing over filtering is at most a factor of 2 in mean-squared estimation uncertainty.
- For unstable observable dynamic system models, the improvement of smoothing over filtering can be orders of magnitude. This "improvement in the improvement" for unstable systems is due, in part, to the smoother using (in effect) a filter in reverse time, for which the effective dynamic system model is stable.

5.5.3 Sources for Additional Information

For historical and technical accounts of seminal works in optimal smoothing, see the surveys by Meditch [188], Kailath [132], and Park and Kailath [211]. The survey by Meditch includes many of the better-known fixed-lag smoother implementation methods.

For a summary of additional smoothing methods and concise technical overviews of smoothing algorithms and their history of development, see the survey by McReynolds [187].

PROBLEMS

5.1 For a scalar system wtih $\Phi = 1$, $H = 1$, $R = 2$ and $Q = 1$, find the steady-state value of the covariance P for a fixed-point smoother estimate.

5.2 Find the solution to Problem 5.1 for P_{10} when $P_0 = 1$, using the fixed-interval smoother on the interval $0 \leq k \leq 10$.

5.3 Solve Problem 5.1 with a fixed-lag smoother with the lag equal to two time steps.

5.4 Repeat Problem 5.1 with $R = 15$.

5.5 Let the model parameters of a Kalman filter be

$$\Phi_{[f]} = \begin{bmatrix} 0.9 & 0.3 \\ -0.3 & 0.9 \end{bmatrix} \tag{5.60}$$

$$H_{[f]} = [\,1 \quad 0\,] \tag{5.61}$$

$$Q_{[f]} = \begin{bmatrix} 0.25 & 0 \\ 0 & 0.25 \end{bmatrix} \tag{5.62}$$

$$R_{[f]} = 1. \tag{5.63}$$

For a compatible BMFLS implementation with a lag of $\ell =$ three time steps:

(a) What is the dimension of $\hat{x}_{[s]}$, the smoother state vector?

(b) Write down the corresponding smoother state-transition matrix $\Phi_{[s]}$

(c) Write down the corresponding smoother measurement sensitivity matrix $H_{[s]}$.

(d) Write down the corresponding smoother disturbance noise covariance matrix $Q_{[s]}$

5.6 For the 2×2 matrix $\Phi_{[f]}$ of Equation 5.60,

(a) Use the MATLAB function `eigs` to find its eigenvalues.

(b) Find their magnitudes using `abs`. Are they <1, 1, or >1?

(c) What can you say about the eigenvalues of the coefficient matrix $F_{[f]}$ for the analogous model in continuous time? Are their real parts positive (i.e., in the right half-plane) or negative (in the left half-plane)?

(d) Is the dynamic system modeled as $x_k = \Phi_{[f]} x_{k-1}$ stable or unstable? Why?

(e) Let

$$x_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

Using MATLAB, generate and plot the components of

$$x_k = \Phi\,[f]^k\, x_0 \quad \text{for} \quad k = 1, 2, 3, \ldots, 100.$$

Do they spiral inward toward the origin or outward toward ∞? How would you interpret that behavior?

(f) Will any fixed-lag smoother based on the same Kalman filter model produce more than a factor of 2 improvement in mean-squared estimation uncertainty? Justify your answer.

5.7 Repeat the exercises of the previous problem with

$$\Phi_{[f]} = \begin{bmatrix} 0.9 & 0.9 \\ -0.9 & 0.9 \end{bmatrix}. \tag{5.64}$$

5.8 Use the parameter values of Problem 5.5 and the MATLAB function `BMFLS` on the CD to demonstrate fixed-lag smoothing on simulated measurements. Remember to use the MATLAB code

```
x = Phi*x + [sqrt(Q(1,1))*randn, sqrt(Q(2,2))*randn];
z = H*x + [sqrt(R(1,1))*randn, sqrt(R(2,2))*randn];
```

to simulate the dynamic noise and measurement noise. Plot

(a) The simulated state vector.

(b) The filter estimate.

(c) The smoother estimate.

(d) RMS filter uncertainties.

(e) RMS smoother uncertainties.

Compare your results to those shown in Figure 5.11 of Example 5.1.

5.9 Use the parameter values of Problem 5.7 and the MATLAB function `BMFLS` on the CD to demonstrate fixed-lag smoothing on simulated measurements. Perform the same tasks as in Problem 5.8.

6

IMPLEMENTATION METHODS

There is a great difference between theory and practice.

—Giacomo Antonelli (1806–1876)[1]

6.1 CHAPTER FOCUS

Up to this point, we have discussed what Kalman filters are and how they are supposed to behave. Their theoretical performance has been shown to be characterized by the covariance matrix of estimation uncertainty, which is computed as the solution of a matrix Riccati differential equation or difference equation.

However, soon after the Kalman filter was first implemented on computers, it was discovered that the observed mean-squared estimation errors were often much larger than the values predicted by the covariance matrix, even with simulated data. The variances of the filter estimation errors were observed to diverge from their theoretical values, and the solutions obtained for the Riccati equation were observed to have negative variances, an embarrassing example of a theoretical impossibility. The problem was eventually determined to be caused by computer roundoff, and alternative implementation methods were developed for dealing with it.

[1]In a letter to the Austrian ambassador, as quoted by Lytton Strachey in *Eminent Victorians* [259]. Cardinal Antonelli was addressing the issue of papal infallibility, but the same might be said about the infallibility of numerical processing systems.

Kalman Filtering: Theory and Practice using MATLAB®, Third Edition. By Mohinder S. Grewal and Angus P. Andrews

This chapter is primarily concerned with

1. how computer roundoff can degrade Kalman filter performance,
2. alternative implementation methods that are more robust against roundoff errors, and
3. the relative computational costs of these alternative implementations.

6.1.1 Main Points to Be Covered

The main points to be covered in this chapter are the following:

1. Computer roundoff errors can and do seriously degrade the performance of Kalman filters.
2. Solution of the matrix Riccati equation is a major cause of numerical difficulties in the conventional Kalman filter implementation, from the standpoint of computational load as well as from the standpoint of computational errors.
3. Unchecked error propagation in the solution of the Riccati equation is a major cause of degradation in filter performance.
4. Asymmetry of the covariance matrix of state estimation uncertainty is a symptom of numerical degradation and a cause of numerical instability, and measures to symmetrize the result can be beneficial.
5. Numerical solution of the Riccati equation tends to be more robust against roundoff errors if *Cholesky factors* or *modified Cholesky factors* of the covariance matrix are used as the dependent variables.
6. Numerical methods for solving the Riccati equation in terms of Cholesky factors are called *factorization methods*, and the resulting Kalman filter implementations are collectively called *square-root filtering*.
7. Information filtering is an alternative state vector implementation that improves numerical stability properties. It is especially useful for problems with very large initial estimation uncertainty.

6.1.2 Topics Not Covered

1. *Parametric Sensitivity Analysis.* The focus here is on numerically stable implementation methods for the Kalman filter. Numerical analysis of *all* errors that influence the performance of the Kalman filter would include the effects of errors on the assumed values of all model parameters, such as Q, R, H, and Φ. These errors also include truncation effects due to finite precision. The sensitivities of performance to these types of modeling errors can be modeled mathematically, but this is not done here.
2. *Smoothing Implementations.* There have been significant improvements in smoother implementation methods beyond those presented in Chapter 5. The interested reader is referred to the surveys by Meditch [188] (methods

up to 1973) and McReynolds [187] (up to 1990) and to earlier results by Bierman [32] and by Watanabe and Tzafestas [281].

3. *Parallel Computer Architectures for Kalman Filtering.* The operation of the Kalman filter can be speeded up, if necessary, by performing some operations in parallel. The algorithm listings in this chapter indicate those loops that can be performed in parallel, but no serious attempt is made to define specialized algorithms to exploit concurrent processing capabilities. An overview of theoretical approaches to this problem is presented by Jover and Kailath [119].

6.2 COMPUTER ROUNDOFF

Roundoff errors are a side effect of computer arithmetic using fixed- or floating-point data words with a fixed number of bits. Computer roundoff is a fact of life for most computing environments.

Example 6.1 (Roundoff Errors) In binary representation, the rational numbers are transformed into sums of powers of 2 as follows:

$$\begin{aligned} 1 &= 2^0 \\ 3 &= 2^0 + 2^1 \\ \frac{1}{3} &= \frac{1}{4} + \frac{1}{16} + \frac{1}{64} + \frac{1}{256} + \cdots \\ &= 0_{\mathrm{b}}0101010101010101010101010\ldots, \end{aligned}$$

where the subscript "b" represents the "binary point" in binary representation (so as not to be confused with the "decimal point" in decimal representation). When 1 is divided by 3 in an ANSI/IEEE standard [14] single-precision floating-point arithmetic, the 1 and the 3 can be represented precisely, but their ratio cannot. The binary representation is limited to 24 bits of mantissa.[2] The above result is then rounded to the 24-bit approximation (starting with the leading 1):

$$\begin{aligned} \frac{1}{3} &\approx 0_{\mathrm{b}}01010101010101010101011 \\ &= \frac{11184811}{33554432} \\ &= \frac{1}{3} - \frac{1}{100663296}, \end{aligned}$$

[2]The mantissa is the part of the binary representation starting with the leading nonzero bit. Because the leading significant bit is always a 1, it can be omitted and replaced by the sign bit. Even including the sign bit, there are effectively 24 bits available for representing the magnitude of the mantissa.

giving an approximation error magnitude of about 10^{-8} and a relative approximation error of about 3×10^{-8}. The difference between the true value of the result and the value approximated by the processor is called *roundoff error*.

6.2.1 Unit Roundoff Error

Computer roundoff for floating-point arithmetic is often characterized by a single parameter $\varepsilon_{\text{roundoff}}$, called the *unit roundoff error*, and defined in different sources as the largest number such that either

$$1 + \varepsilon_{\text{roundoff}} \equiv 1 \text{ in machine precision} \tag{6.1}$$

or

$$1 + \varepsilon_{\text{roundoff}}/2 \equiv 1 \text{ in machine precision.} \tag{6.2}$$

The name "eps" in MATLAB is the parameter satisfying the second of these equations. Its value may be found by typing "eps⟨RETURN⟩" (i.e., typing "eps" without a following semicolon, followed by hitting the RETURN or ENTER key) in the MATLAB command window. Entering "−log2(eps)" should return the number of bits in the mantissa of the standard data word.

6.2.2 Effects of Roundoff on Kalman Filter Performance

6.2.2.1 Early Discoveries Around the time the Kalman filter was introduced in 1960, the International Business Machines Corporation (IBM) had introduced its 700-series computers with 36-bit floating-point arithmetic units. This was considered quite revolutionary at the time but, even with all this precision, computer roundoff in the first Kalman filter implementations was a serious problem. Early accounts by McGee and Schmidt [186], Schmidt [241], and Battin [25] emphasize the difficulties encountered with computer roundoff and the serious risk of failure it entailed for in-flight implementations in the Apollo moon missions. The problem was eventually solved satisfactorily by finding new implementation methods, and even better methods would be discovered after the last Apollo missions in the early 1970s.

Many of these roundoff problems occurred on computers with much shorter word-lengths than those available for modern MATLAB implementations and with less accurate implementations of bit-level arithmetic than current microprocessors adhering to current ANSI/IEEE standards [14].

However, the next example (modified after one from [78], credited to Richard J. Hanson) demonstrates that roundoff can still be a problem in Kalman filter implementations in MATLAB environments and how a problem that is well conditioned, as posed, can be made ill-conditioned by the filter implementation.

Example 6.2 (Comparing Different Implementation Methods) Let I_n denote the $n \times n$ identity matrix. Consider the filtering problem with measurement sensitivity

matrix

$$H = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1+\delta \end{bmatrix}$$

and covariance matrices

$$P_0 = I_3 \quad \text{and} \quad R = \delta^2 I_2$$

where $\delta^2 < \varepsilon_{\text{roundoff}}$ but $\delta > \varepsilon_{\text{roundoff}}$. In this case, although H clearly has rank $= 2$ in machine precision, the product $HP_0\mathrm{H}^{\mathrm{T}}$ with roundoff will equal

$$\begin{bmatrix} 3 & 3+\delta \\ 3+\delta & 3+2\delta \end{bmatrix},$$

which is singular. The result is unchanged when R is added to HP_0H^{T}. In this case, then, the filter observational update fails because the matrix $HP_0H^{\mathrm{T}} + \mathrm{R}$ is not invertible.

Sneak Preview of Alternative Implementations Figure 6.1 illustrates how the standard Kalman filter and some of the alternative implementation methods perform on the variably ill-conditioned problem of Example 6.2 (implemented as the MATLAB m-file `shootout.m` on the accompanying CD) as the conditioning parameter $\delta \to 0$. All solution methods were implemented in the same precision (64-bit floating point) in MATLAB. The labels on the curves in this plot correspond to the names of the corresponding m-file implementations on the accompanying CD.

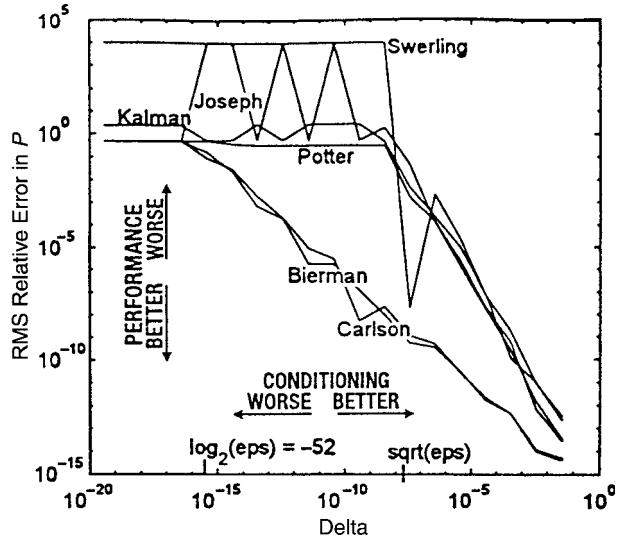


Figure 6.1 Degradation of Riccati equation observational updates with problem conditioning.

These are also the names of the authors of the corresponding methods, the details of which will be presented further on.

For this particular example, the accuracies of the methods labeled "Carlson" and "Bierman" appear to degrade more gracefully than the others as $\delta \rightarrow \varepsilon$, the machine precision limit. The Carlson and Bierman solutions still maintain about 9 digits ($\approx$30 bits) of accuracy at $\delta \approx \sqrt{\varepsilon}$ when the other methods have essentially no bits of accuracy in the computed solution.

This one example, by itself, does not prove the general superiority of the Carlson and Bierman solutions for the observational updates of the Riccati equation. The full implementation will require a compatible method for performing the temporal update as well. However, the observational update had been the principal source of difficulty with the conventional implementation.

6.2.3 Terminology of Numerical Error Analysis

We first need to define some general terms used in characterizing the influence of roundoff errors on the accuracy of the numerical solution to a given computation problem.

Robustness and Numerical Stability These terms are used to describe qualitative properties of arithmetic problem-solving methods. *Robustness* refers to the relative insensitivity of the solution to errors of some sort. *Numerical stability* refers to robustness against roundoff errors.

Precision versus Numerical Stability Relative roundoff errors can be reduced by using more precision (i.e., more bits in the mantissa of the data format), but the accuracy of the result is also influenced by the accuracy of the initial parameters used and the procedural details of the implementation method. Mathematically equivalent implementation methods can have very different numerical stabilities at the same precision.

Numerical Stability Comparisons Numerical stability comparisons can be slippery. Robustness and stability of solution methods are matters of degree, but implementation methods cannot always be totally ordered according to these attributes. Some methods are considered more robust than others, but their relative robustness can also depend upon intrinsic properties of the problem being solved.

III-Conditioned and Well-Conditioned Problems In the analysis of numerical problem-solving methods, the qualitative term *conditioning* is used to describe the sensitivity of the error in the output (solution) to variations in the input data (problem). This sensitivity generally depends on the input data and the solution method.

A problem is called *well-conditioned* if the solution is not "badly" sensitive to the input data and *ill-conditioned* if the sensitivity is "bad." The definition of what is bad generally depends on the uncertainties of the input data and the numerical precision being used in the implementation. One might, for example, describe a matrix A as

being "ill-conditioned with respect to inversion" if A is "close" to being singular. The definition of "close" in this example could mean within the uncertainties in the values of the elements of A or within machine precision.

Example 6.3 (Condition Number of a Matrix) The sensitivity of the solution x of the linear problem $Ax = b$ to uncertainties in the input data (A and b) and roundoff errors is characterized by the *condition number* of A, which can be defined as the ratio

$$cond(A) = \frac{\max_x ||Ax||/||x||}{\min_x ||Ax||/||x||} \tag{6.3}$$

if A is nonsingular and as ∞ if A is singular. It also equals the ratio of the largest and smallest characteristic values of A. Note that the condition number will always be ≥ 1 because max $\geq$ min. As a general rule in matrix inversion, condition numbers close to 1 are a good omen, and increasingly larger values are cause for increasing concern over the validity of the results.

The *relative error* in the computed solution $\hat{x}$ of the equation $Ax = b$ is defined as the ratio $\|\hat{x} - x\|/\|x\|$ of the magnitude of the error to the magnitude of x.

As a rule of thumb, the maximum relative error in the computed solution is bounded above by $c_A \varepsilon_{\text{roundoff}}$ cond(A), where $\varepsilon_{\text{roundoff}}$ is the unit roundoff error in computer arithmetic (defined in Section 6.2.1) and the positive constant c_A depends on the dimension of A. The problem of computing x, given A and b, is considered ill-conditioned if adding 1 to the condition number of A in computer arithmetic has no effect. That is, the logical expression $1 + \text{cond}(A) = \text{cond}(A)$ evaluates to true.

Consider an example with the coefficient matrix

$$A = \begin{bmatrix} 1 & L & 0 \\ 0 & 1 & L \\ 0 & 0 & 1 \end{bmatrix},$$

where

$$\begin{aligned} L &= 2^{64} \\ &= 18{,}446{,}744{,}073{,}709{,}551{,}616 \end{aligned}$$

which is such that computing L^2 would cause overflow in ANSI/IEEE standard single-precision arithmetic.

The condition number of A will then be

$$\text{cond}(A) \approx 3.40282 \times 10^{38}.$$

This is about 31 orders of magnitude beyond where the rule-of-thumb test for ill-conditioning would fail in this precision ($\approx 2 \times 10^7$). One would then consider A extremely ill-conditioned for inversion (which it is) even though its determinant equals 1.

PROGRAMMING NOTE For the general linear equation problem $Ax = b$, it is not necessary to invert A explicitly in the process of solving for x, and numerical stability is generally improved if matrix inversion is avoided. The MATLAB matrix divide (using `x = A\b`) does this.

6.2.4 Ill-Conditioned Kalman Filtering Problems

For Kalman filtering problems, the solution of the associated Riccati equation should equal the covariance matrix of the actual estimation uncertainty, which should be optimal with respect to all quadratic loss functions. The computation of the Kalman (optimal) gain depends on it. If this does not happen, the problem is considered ill-conditioned. Factors that contribute to such ill-conditioning include the following:

1. Large uncertainties in the values of the matrix parameters Φ, Q, H, or R. Such modeling errors are not accounted for in the derivation of the Kalman filter.
2. Large ranges of the actual values of these matrix parameters, the measurements, or the state variables—all of which can result from poor choices of scaling or dimensional units.
3. Ill-conditioning of the intermediate result $R^* = HPH^{\mathrm{T}} + R$ for inversion in the Kalman gain formula.
4. Ill-conditioned theoretical solutions of the matrix Riccati equation—without considering numerical solution errors. With numerical errors, the solution may become indefinite, which can destabilize the filter estimation error.
5. Large matrix dimensions. The number of arithmetic operations grows as the square or cube of matrix dimensions, and each operation can introduce roundoff errors.
6. Poor machine precision, which makes the relative roundoff errors larger.

Some of these factors are unavoidable in many applications. Keep in mind that they do not *necessarily* make the Kalman filtering problem hopeless. However, they are cause for concern—and for considering alternative implementation methods.

6.3 EFFECTS OF ROUNDOFF ERRORS ON KALMAN FILTERS

Quantifying the Effects of Roundoff Errors on Kalman Filtering Although there was early experimental evidence of divergence due to roundoff errors, it has been difficult to obtain general principles describing how it is related to characteristics of the implementation. There are some general (but somewhat weak) principles relating roundoff errors to characteristics of the computer on which the filter is implemented and to properties of the filter parameters. These include the results of Verhaegen and Van Dooren [278] on the numerical analysis of various implementation methods in Kalman filtering. These results provide upper bounds on the propagation of roundoff errors as functions of the norms and singular values of key matrix variables. They

show that some implementations have better bounds than others. In particular, they show that certain *symmetrization* procedures are provably beneficial and that the so-called square-root filter implementations have generally better error propagation bounds than the conventional Kalman filter equations.

Let us examine the ways that roundoff errors propagate in the computation of the Kalman filter variables and how they influence the accuracy of results in the Kalman filter. Finally, we provide some examples that demonstrate common failure modes.

6.3.1 Roundoff Error Propagation in Kalman Filters

Heuristic Analysis We begin with a heuristic look at roundoff error propagation, from the viewpoint of the data flow in the Kalman filter, to show how roundoff errors in the Riccati equation solution are not controlled by feedback like roundoff errors in the estimate. Consider the matrix-level data flow diagram of the Kalman filter shown in Figure 6.2. This figure shows the data flow at the level of vectors

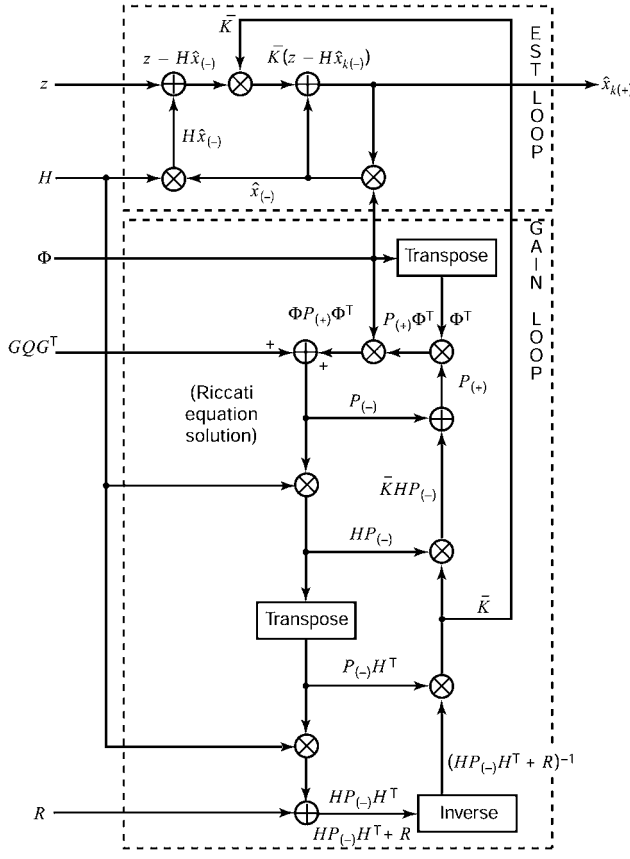


Figure 6.2 Kalman filter data flow.

and matrices, with operations of addition ($\oplus$), multiplication ($\otimes$), and inversion ($I\div$). Matrix transposition need not be considered a data operation in this context, because it can be implemented by index changes in subsequent operations. This data flow diagram is fairly representative of the straightforward Kalman filter algorithm, the way it was originally presented by Kalman, and as it might be implemented in MATLAB by a moderately conscientious programmer. That is, the diagram shows how partial results (including the Kalman gain, $\overline{K}$) might be saved and reused. Note that the internal data flow can be separated into two semi-independent loops within the dashed boxes. The variable propagated around one loop is the state estimate. The variable propagated around the other loop is the covariance matrix of estimation uncertainty. (The diagram also shows some of the loop "shortcuts" resulting from reuse of partial results, but the basic data flows are still loops.)

Feedback in the Estimation Loop The uppermost of these loops, labeled EST. LOOP, is essentially a feedback error correction loop with gain ($\overline{K}$) computed in the other loop (labeled GAIN LOOP). The difference between the expected value $H\hat{x}$ of the observation z (based on the current estimate $\hat{x}$ of the state vector) and the observed value is used in correcting the estimate $\hat{x}$. Errors in $\hat{x}$ will be corrected by this loop so long as the gain is correct. This applies to errors in $\hat{x}$ introduced by roundoff as well as those due to noise and *a priori* estimation errors. Therefore, roundoff errors in the estimation loop are compensated for by the feedback mechanism, so long as the loop gain is correct. That gain is computed in the other loop.

No Feedback in the Gain Loop This is the loop in which the Riccati equation is solved for the covariance matrix of estimation uncertainty (P), and the Kalman gain is computed as an intermediate result. It is not stabilized by feedback, the way that the estimation loop is stabilized. There is no external reference for correcting the "estimate" of P. Consequently, there is no way of detecting and correcting the effects of roundoff errors. They propagate and accumulate unchecked. This loop also includes many more roundoff operations than the estimation loop, as evidenced by the greater number of matrix multiplies ($\otimes$) in the loop. The computations involved in evaluating the filter gains are, therefore, more suspect as sources of roundoff error propagation in this "conventional" implementation of the Kalman filter. It has been shown by Potter [216] that the gain loop, by itself, is not unstable. However, even bounded errors in the computed value of P may momentarily destabilize the estimation loop.

Example 6.4 (Effects of Negative Characteristic Values on the Estimation Errors) Roundoff errors can cause the computed value of P to have a negative characteristic value. The Riccati equation is stable, and the problem will eventually rectify itself. However, the effect on the actual estimation error can be a more serious problem.

Because P is a factor in the Kalman gain $\overline{K}$, a negative characteristic value of P can cause the gain in the prediction error feedback loop to have the wrong sign. However, in this transient condition, the estimation loop is momentarily destabilized. In this illustration, the estimate $\hat{x}$ converges toward the true value x until the gain changes

sign. Then the error diverges momentarily. The gain computations may eventually recover with the correct sign, but the accumulated error due to divergence is not accounted for in the gain computations. The gain is not as big as it should be, and convergence is slower than it should be.

6.3.1.1 Numerical Analysis Because the *a priori* value of P is the one used in computing the Kalman gain, it suffices to consider just the error propagation of that value. It is convenient, as well, to consider the roundoff error propagation for $x(-)$.

A first-order roundoff error propagation model is of the form

$$\delta x_{k+1}(-) = f_1(\delta x_k(-), \delta P_k(-)) + \Delta x_{k+1}, \tag{6.4}$$

$$\delta P_{k+1}(-) = f_2(\delta P_k(-) + \Delta P_{k+1}(-), \tag{6.5}$$

where the δ term refers to the accumulated error and the Δ term refers to the added roundoff errors on each recursion step. This model ignores higher order terms in the error variables. The forms of the appropriate error propagation functions are given in Table 6.1. Error equations for the Kalman gain are also given, although the errors in $\overline{K}_k$ depend only on the errors in x and P—they are not propagated independently. These error propagation function values are from the paper by Verhaegen and Van Dooren [278]. (Many of these results have also appeared in earlier publications.) These expressions represent the first-order error in the updated *a priori* variables on the $(k+1)$th temporal epoch in terms of the first-order errors in the kth temporal epoch and the errors added in the update process.

Roundoff Error Propagation Table 6.1 compares two filter implementation types in terms of their first-order error propagation characteristics. One implementation type is called *conventional*. That corresponds to the straightforward implementation of the equations as they were originally derived in previous chapters, excluding the "Joseph-stabilized" implementation mentioned in Chapter 4. The other type is called *square root*, the type of implementation presented in this chapter. A further breakdown of these implementation types will be defined in later sections.

TABLE 6.1 First-Order Error Propagation Models

Roundoff Error in Filter Variable	Error Model (by Filter Type)	
	Conventional Implementation	Square-Root Covariance
$\delta x_{k+1}(-)$	$A_1[\delta x_k(-) + \delta P_k(-)A_2(z - Hx_k(-))] + \Delta x_{k+1}$	
$\delta \overline{K}_k$	$A_1 \delta P_k(-)$	
$\delta P_{k+1}(-)$	$A_1 \delta P_k(-) A_1^{\mathrm{T}} + \Delta P_{k+1}$ $+ \Phi(\delta P_k(-) - \delta P_k^{\mathrm{T}}(-))\Phi^{\mathrm{T}}$ $- \Phi(\delta P_k(-) - \delta P_k^{\mathrm{T}}(-))A_1^{\mathrm{T}}$	$A_1 \delta P_k(-) A_1^{\mathrm{T}}$ $+ \Delta P_{k+1}$

Note: $A_1 = \Phi - \overline{K}_k H$; $A_2 = H^{\mathrm{T}}[HP_kH^{\mathrm{T}} + R]^{-1}$.

Propagation of Antisymmetry Errors Note the two terms in Table 6.1 involving the antisymmetry error $\delta P_k(-) - \delta P_k^{\mathrm{T}}(-)$ in the covariance matrix P, which tends to confirm in theory what had been discovered in practice. Early computers had very little memory capacity, and programmers had learned to save time and memory by computing only the unique parts of symmetric matrix expressions such as $\Phi P \Phi^{\mathrm{T}}$, HPH^{T}, $HPH^{\mathrm{T}} + R$, or $(HPH^{\mathrm{T}} + R)^{-1}$. To their surprise and delight, this was also found to improve error propagation. It has also been found to be beneficial in MATLAB implementations to maintain symmetry of P by evaluating the MATLAB expression `P = .5*(P + P')` on every cycle of the Riccati equation.

Added Roundoff Error The roundoff error (Δ) that is added on each cycle of the Kalman filter is considered in Table 6.2. The tabulated formulas are upper bounds on these random errors.

The important points which Table 6.2 demonstrates are the following:

1. These expressions show the same first-order error propagation in the state update errors for both filter types (covariance and square-root forms). These include terms coupling the errors in the covariance matrix to the state estimate and gain.
2. The error propagation expression for the conventional Kalman filter includes aforementioned terms proportional to the antisymmetric part of P. One must consider the effects of roundoff errors added in the computation of x, $\overline{K}$, and P, as well as those propagated from the previous temporal epoch. In this case, Verhaegen and Van Dooren [278] have obtained upper bounds on the norms of the added errors $\Delta x, \Delta\overline{K}$, and ΔP, as shown in Table 6.2. These upper bounds give a crude approximation of the dependence of roundoff error propagation on the characteristics of the unit roundoff error (ε) and the

TABLE 6.2 Upper Bounds on Added Roundoff Errors

	Upper Bounds (by Filter Type)	
Norm of Roundoff Errors	Conventional Implementation	Square-Root Covariance
$\lvert\Delta x_{k+1}(-)\rvert$	$\varepsilon_1(\lvert A_1\rvert\lvert x_k(-)\rvert + \lvert\overline{K}_k\rvert\lvert z_k\rvert) + \lvert\Delta\overline{K}_k\rvert(\lvert H\rvert\lvert x_k(-)\rvert + \lvert z_k\rvert)$	$\varepsilon_4(\lvert A_1\rvert\lvert x_k(-)\rvert + \lvert\overline{K}_k\rvert\lvert z_k\rvert) + \lvert\Delta\overline{K}_k\rvert(\lvert H\rvert\lvert x_k(-)\rvert + \lvert z_k\rvert)$
$\lvert\Delta\overline{K}_k\rvert$	$\varepsilon_2\kappa^2(R^*)\lvert\overline{K}_k\rvert$	$\varepsilon_5\kappa(R^*)[\lambda_m^{-1}(R^*)\lvert C_{p(\overline{k}+1)}\rvert + \lvert\overline{K}_k C_{R^*}\rvert + \lvert A_3\rvert/\lambda_1(R^*)]$
$\lvert\Delta P_{k+1}(-)\rvert$	$\varepsilon_3\kappa^2(R^*)\lvert P_{k+1}(-)\rvert$	$\dfrac{\varepsilon_6[1+\kappa(R^*)]\lvert P_{k+1}\rvert\lvert A_3\rvert}{\lvert C_{p(k+1)}\rvert}$

Note: $\varepsilon_1, \ldots, \varepsilon_6$ are constant multiples of ε, the unit roundoff error;
$A_1 = \Phi - \overline{K}_k H$; $A_3 = [(\overline{K}_k C_{R^*})\lvert C_{p(k+1)}\rvert]$; $R^* = HP_k(-)H^{\mathrm{T}} + R$; $R^* = C_{R^*} C_{R^*}^{\mathrm{T}}$ (triangular Cholesky decomposition);
$P_{k+1}(-) = C_{p(k+1)} C_{p(k+1)}^{\mathrm{T}}$ (triangular Cholesky decomposition);
$\lambda_1(R^*) \geq \lambda_2(R^*) \geq \cdots \geq \lambda_m(R^*) \geq 0$ are the characteristic values of R^*;
$\kappa(R^*) = \lambda_1(R^*)/\lambda_m(R^*)$ is the condition number of R^*.

parameters of the Kalman filter model. Here, the bounds on the added state estimation error are similar for the two filter types, but the bounds on the added covariance error ΔP are better for the square-root filter. (The factor is something like the condition number of the matrix R^*.) In this case, one cannot relate the difference in performance to such factors as asymmetry of P.

The efficacy of various implementation methods for reducing the effects of roundoff errors have also been studied experimentally for some applications. The paper by Verhaegen and Van Dooren [278] includes results of this type as well as numerical analyses of other implementations (information filters and Chandrasekhar filters). Similar comparisons of square-root filters with conventional Kalman filters (and Joseph-stabilized filters) have been made by Thornton and Bierman [269].

6.3.2 Examples of Filter Divergence

The following simple examples show how roundoff errors can cause the Kalman filter results to diverge from their expected values.

Example 6.5 (Roundoff Errors Due to Large *a Priori* Uncertainty) If users have very little confidence in the *a priori* estimate for a Kalman filter, they tend to make the initial covariance of estimation uncertainty very large. This has its limitations, however.

Consider the scalar parameter estimation problem ($\Phi = I$, $Q = 0$, $\ell = n = 1$), in which the initial variance of estimation uncertainty $P_0 \gg R$, the variance of measurement uncertainty. Suppose that the measurement sensitivity $H = 1$ and that P_0 is so much greater than R that, in floating-point machine precision, the result of adding R to P_0—with roundoff—is P_0. That is, $R < \varepsilon P_0$. In that case, the values computed in the Kalman filter calculations will be as shown in the following table:

Observation Number	Expression	Value	
		Exact	Rounded
1	$P_0 H^{\mathrm{T}}$	P_0	P_0
1	HP_0H^{T}	P_0	P_0
1	$HP_0H^{\mathrm{T}} + R$	$P_0 + R$	P_0
1	$\overline{K}_1 = P_0H^{\mathrm{T}}(HP_0H^{\mathrm{T}} + R)^{-1}$	$\dfrac{P_0}{P_0 + R}$	1
1	$P_1 = P_0 - \overline{K}_1 HP_0$	$\dfrac{P_0R}{P_0 + R}$	0
⋮	⋮	⋮	⋮
k	$\overline{K}_k = P_{k-1}H^{\mathrm{T}}(HP_{k-1}H^{\mathrm{T}} + R)^{-1}$	$\dfrac{P_0}{kP_0 + R}$	0
	$P_k = P_{k-1} - \overline{K}_k HP_{k-1}$	$\dfrac{P_0R}{kP_0 + R}$	0

The rounded value of the calculated variance of estimation uncertainty is *zero* after the first measurement update and remains zero thereafter. As a result, the calculated value of the Kalman gain is also zero after the first update. The exact (roundoff-free) value of the Kalman gain is $\approx 1/k$, where k is the observation number. After 10 observations,

1. the calculated variance of estimation uncertainty is zero;
2. the actual variance of estimation uncertainty is $P_0R/(P_0 + R) \approx R$ (the value after the first observation and after which the computed Kalman gains were zeroed), and
3. the theoretical variance in the exact case (no roundoff) would have been $P_0R/(10P_0 + R) \approx \frac{1}{10}R$.

The ill-conditioning in this example is due to the misscaling between the *a priori* state estimation uncertainty and the measurement uncertainty.

6.4 FACTORIZATION METHODS FOR SQUARE-ROOT FILTERING

6.4.1 James E. Potter and Square-Root Filtering

The most reliable and numerical stable implementations of the Kalman filter are collectively called *square-root filters.* The fundamental concept for all these methods is to reformulate the matrix Riccati equation in terms of certain factors of the covariance matrix P of estimation uncertainty. These factors are generally much better conditioned for matrix operations than P itself. As a consequence, the implementations using these factors are generally much more robust against roundoff errors.

The idea of transforming the Riccati equation in this way originated with James E. Potter (1937–2005) in 1962, when he was a graduate student in mathematics at MIT and working part-time at the MIT Instrumentation Laboratory on the Apollo Project to take Americans to the moon and back. The space navigation problem for Apollo was the first real-world application of the Kalman filter, and there was a serious problem with its intended implementation on the Apollo computer. The critical problem was poor numerical stability of the Riccati equation solution, especially the measurement updates.

Potter took the problem home with him one Friday afternoon and returned on Monday morning with the solution. He had recast the measurement update equations of the Riccati equation in terms of Cholesky factors and *elementary matrices* of the type introduced by Householder [112]. Potter found that he could factor an elementary matrix into a product of its matrix square roots, and that led to a formula for the measurement update of a Cholesky factor of the covariance matrix. The key factoring formula in Potter's derivation involved taking square roots of elementary matrices, and Potter's implementation came to be called *square-root filtering*. This innovation would contribute enormously to the success of the first major Kalman filtering

application. Potter's square-root filter was implemented on the Apollo computer (in 14-bit arithmetic) and operated successfully on all the Apollo missions.

Alternative square-root filter implementations with reduced computational complexity were developed within the next decade or so. All these implementations rely on matrix factorization methods, some of which are called *triangularization*. Numerically stable methods for factoring matrices are described in Sections B.6 and 6.4.3. This section describes how these methods are applied to Kalman filtering.

6.4.2 Overview of Matrix Factorization Tricks

Matrix Factoring and Decomposition The terms *decomposition* and *factoring* (or *factorization*) are used interchangeably to describe the process of transforming a matrix or matrix expression into an equivalent product of factors.[3]

Applications to Kalman Filtering The more numerically stable implementations of the Kalman filter use one or more of the following techniques to solve the associated Riccati equation:

1. Factoring the *covariance matrix of state estimation uncertainty P* (the dependent variable of the Riccati equation) into Cholesky factors (see Section B.6) or into modified Cholesky factors (unit triangular and diagonal factors).
2. Factoring the *covariance matrix of measurement noise R* to reduce the computational complexity of the observational update implementation. (These methods effectively "decorrelate" the components of the measurement noise vector.)
3. Taking the symmetric *matrix square roots of elementary matrices*. A symmetric elementary matrix has the form $I - \sigma vv^{\mathrm{T}}$, where I is the $n \times n$ identity matrix, σ is a scalar, and v is an n-vector. The symmetric square root of an elementary matrix is also an elementary matrix with the same v but a different value for σ.
4. Factoring *general matrices* as products of triangular and orthogonal matrices. Two general methods are used in Kalman filtering:
   a. *Triangularization (QR decomposition)* methods were originally developed for more numerically stable solutions of systems of linear equations. They factor a matrix into the product of an orthogonal matrix Q and a triangular matrix R. In the application to Kalman filtering, only the triangular factor is needed. We will call the QR decomposition *triangularization*, because Q

[3]The term *decomposition* is somewhat more general. It is also used to describe nonproduct representations, such as the *additive decomposition* of a square matrix into its symmetric and antisymmetric parts:

$$A = \tfrac{1}{2}(A + A^{\mathrm{T}}) + \tfrac{1}{2}(A - A^{\mathrm{T}}).$$

Another distinction between *decomposition* and *factorization* is made by Dongarra et al. [75], who use the term *factorization* to refer to an arithmetic process for performing a product decomposition of a matrix in which not all factors are preserved. The term *triangularization* is used in this book to indicate a QR factorization (in the sense of Dongarra et al.) involving a triangular factor that is preserved and an orthogonal factor that is not preserved.

and R already have special meanings in Kalman filtering. The two triangularization methods used in Kalman filtering are:

i. *Givens rotations* [97] triangularize a matrix by operating on one element at a time. (A *modified Givens method* due to Gentleman [96] generates *diagonal* and *unit triangular* factors.)

ii. *Householder transformations* triangularize a matrix by operating on one row or column at a time.

b. *Gram–Schmidt orthonormalization* is another general method for factoring a general matrix into a product of an orthogonal matrix and a triangular matrix. Usually, the triangular factor is not saved. In the application to Kalman filtering, only the triangular factor is saved.

5. *Rank 1 modification algorithms*. A rank 1 modification of a symmetric positive-definite $n \times n$ matrix M has the form $M \pm vv^{\mathrm{T}}$, where v is an n-vector (and therefore has matrix rank equal to 1). The algorithms compute a Cholesky factor of the modification $M \pm vv^{\mathrm{T}}$, given v and a Cholesky factor of M.

6. *Block matrix factorizations* of matrix expressions in the Riccati equation. The general approach uses two different factorizations to represent the two sides of an equation, such as

$$\begin{aligned} CC^{\mathrm{T}} &= AA^{\mathrm{T}} + BB^{\mathrm{T}} \\ &= \begin{bmatrix} A & B \end{bmatrix} \begin{bmatrix} A^{\mathrm{T}} \\ B^{\mathrm{T}} \end{bmatrix}. \end{aligned}$$

The alternative Cholesky factors C and $[A \quad B]$ must then be related by orthogonal transformations (triangularizations). A QR decomposition of $[A \quad B]$ will yield a corresponding solution of the Riccati equation in terms of a Cholesky factor of the covariance matrix.

In the example used above, $[A \quad B]$ would be called a 1×2 block-partitioned matrix, because there are one row and two columns of blocks (matrices) in the partitioning. Different block dimensions are used to solve different problems:

1. The *discrete-time temporal update* equation is solved in square-root form by using alternative 1×2 block-partitioned Cholesky factors.
2. The *observational update* equation is solved in square-root form by using alternative 2×2 block-partitioned Cholesky factors and modified Cholesky factors representing the observational update equation.
3. The *combined temporal/observational update* equations are solved in square-root form by using alternative 2×3 block-partitioned Cholesky factors of the combined temporal and observational update equations.

The different implementations of the Kalman filter based on these approaches are presented in Sections 6.5.2–6.6.2 and 6.6. They make use of the general numerical procedures presented in Sections 6.4.3–6.4.6.

6.4.3 Cholesky Decomposition Methods and Applications

Symmetric Products and Cholesky Factors The product of a matrix C with its own transpose in the form $CC^{\mathrm{T}} = M$ is called the *symmetric product* of C, and C is called a *Cholesky factor of* M (Section B.6). Strictly speaking, a Cholesky factor is not a matrix square root, although the terms are often used interchangeably in the literature. (A matrix square root S of M is a solution of $M = SS = S^2$ without the transpose.)

All symmetric nonnegative definite matrices (such as covariance matrices) have Cholesky factors, but the Cholesky factor of a given symmetric nonnegative definite matrix is *not unique*. For any *orthogonal matrix* $\mathcal{J}$ (i.e., such that $\mathcal{J}\mathcal{J}^{\mathrm{T}} = I$), the product $\Gamma = C\mathcal{J}$ satisfies the equation

$$\Gamma\Gamma^{\mathrm{T}} = C\mathcal{J}\mathcal{J}^{\mathrm{T}}C^{\mathrm{T}} = CC^{\mathrm{T}} = M.$$

That is, $\Gamma = C\mathcal{J}$ is also a Cholesky factor of M. Transformations of one Cholesky factor into another are important for alternative Kalman filter implementations.

Applications to Kalman Filtering Cholesky decomposition methods produce triangular matrix factors (Cholesky factors), and the sparseness of these factors can be exploited in the implementation of the Kalman filter equations. These methods are used for the following purposes:

1. in the decomposition of covariance matrices (P, R, and Q) for implementation of square-root filters;
2. in decorrelating measurement errors between components of vector-valued measurements so that the components may be processed sequentially as independent scalar-valued measurements (Section 6.4.3.2);
3. as part of a numerically stable method for computing matrix expressions containing the factor $(HPH^{\mathrm{T}} + R)^{-1}$ in the conventional form of the Kalman filter (this matrix inversion can be obviated by the decorrelation methods, however); and
4. in Monte Carlo analysis of Kalman filters by simulation, in which Cholesky factors are used for generating independent random sequences of vectors with prespecified means and covariance matrices (see Section 3.4.8).

6.4.3.1 Cholesky Decomposition Algorithms

Symmetric Cholesky Factors The *singular value decomposition* of a symmetric positive-definite matrix P has the form

$$P = \mathcal{E}\Lambda\mathcal{E}^{\mathrm{T}} \tag{6.6}$$

where

$\mathcal{E}$ = an orthogonal matrix,

Λ = a diagonal matrix with positive diagonal values $\lambda_j > 0$.

The "eigenstructure" of P is characterized by the columns e_j of $\mathcal{E}$ and the diagonal elements λ_j of Λ. The e_j are unit vectors parallel to the principal axes of the equiprobability hyperellipse of the Gaussian distribution with covariance P, and the λ_j are the corresponding variances of the probability distribution along these axes. Furthermore, the values e_j and λ_j define the eigenvector-eigenvalue decomposition of P as

$$P = \sum_{j=1}^{n} \lambda_j e_j e_j^{\mathrm{T}}. \tag{6.7}$$

They also define a symmetric Cholesky factor of P as

$$C \stackrel{\text{def}}{=} \sum_{j=1}^{n} \sqrt{\lambda_j} e_j e_j^{\mathrm{T}}, \tag{6.8}$$

such that $C^{\mathrm{T}}C = CC^{\mathrm{T}} = P$. The first of these equalities follows from the symmetry of C (i.e., $C^{\mathrm{T}} = C$) and the second from

$$CC^{\mathrm{T}} = \left[\sum_{j=1}^{n} \sqrt{\lambda_j} e_j e_j^{\mathrm{T}}\right]\left[\sum_{k=1}^{n} \sqrt{\lambda_k} e_k e_k^{\mathrm{T}}\right]^{\mathrm{T}} \tag{6.9}$$

$$= \sum_{j=1}^{n}\sum_{k=1}^{n} \sqrt{\lambda_j}\sqrt{\lambda_k} e_j \left\{e_j^{\mathrm{T}} e_k\right\} e_k^{\mathrm{T}} \tag{6.10}$$

$$\left\{e_j^{\mathrm{T}} e_k\right\} = \begin{cases} 0, & j \neq k \\ 1, & j = k \end{cases} \tag{6.11}$$

$$CC^{\mathrm{T}} = \sum_{j=1}^{n} \left[\sqrt{\lambda_j}\right]^2 e_j e_j^{\mathrm{T}} \tag{6.12}$$

$$= \sum_{j=1}^{n} \lambda_j e_j e_j^{\mathrm{T}} \tag{6.13}$$

$$= P. \tag{6.14}$$

The MATLAB function `svd` performs singular value decompositions of general matrices. If P is symmetric positive definite, the MATLAB command "`[E,Lambda,X] = svd(P);`" returns the eigenvectors e_j of P as the columns of `E` and the eigenvalues λ_j of P as the diagonal elements of `Lambda`. The symmetric positive definite Cholesky factor C of a symmetric positive definite matrix P can then be produced by MATLAB commands

```
[E,Lambda,X] = svd(P);
sqrtD        = sqrt(Lambda);
C            = E*sqrtD*E';
```

Symmetric Cholesky factors are not very popular in Kalman filtering, because there are much simpler algorithms for computing triangular Choleksy factors, but symmetric Cholesky factors may be useful in sample-based methods for nonlinear filtering.

Triangular Matrices Recall that the *main diagonal* of an $n \times m$ matrix C is the set of elements $\{C_{ii} \mid 1 \leq i \leq \min(m, n)\}$ and that C is called *triangular* if the elements on one side of its main diagonal are zero. The matrix is called *upper triangular* if its nonzero elements are on and above its main diagonal and *lower triangular* if they are on or below its main diagonal.

A Cholesky decomposition algorithm is a procedure for calculating the elements of a triangular Cholesky factor of a symmetric, nonnegative definite matrix. It solves the Cholesky decomposition equation $P = CC^{\mathrm{T}}$ for a triangular matrix C, given the matrix P, as illustrated in the following example.

Example 6.6 (Cholesky Algorithm for 3 × 3 Matrix) Consider the 3×3 example for finding a lower triangular Cholesky factor $P = CC^{\mathrm{T}}$ for symmetric P:

$$\begin{bmatrix} p_{11} & p_{21} & p_{31} \\ p_{21} & p_{22} & p_{32} \\ p_{31} & p_{32} & p_{33} \end{bmatrix} = \begin{bmatrix} c_{11} & 0 & 0 \\ c_{21} & c_{22} & 0 \\ c_{31} & c_{32} & c_{33} \end{bmatrix} \begin{bmatrix} c_{11} & 0 & 0 \\ c_{21} & c_{22} & 0 \\ c_{31} & c_{32} & c_{33} \end{bmatrix}^{\mathrm{T}}$$

$$= \begin{bmatrix} c_{11}^2 & c_{11}c_{21} & c0_{11}c_{31} \\ c_{11}c_{21} & c_{21}^2 + c_{22}^2 & c_{21}c_{31} + c_{22}c_{32} \\ c_{11}c_{31} & c_{21}c_{31} + c_{22}c_{32} & c_{31}^2 + c_{32}^2 + c_{33}^2 \end{bmatrix}.$$

The corresponding matrix elements of the left- and right-hand sides of the last matrix equation can be equated as nine scalar equations. However, due to symmetry, only six of these are independent. The six scalar equations can be solved in sequence, making use of previous results. The following solution order steps down the rows and across the columns:

Six Independent Scalar Equations	Solutions Using Prior Results
$p_{11} = c_{11}^2$	$c_{11} = \sqrt{p_{11}}$
$p_{21} = c_{11}c_{21}$	$c_{21} = p_{21}/c_{11}$
$p_{22} = c_{21}^2 + c_{22}^2$	$c_{22} = \sqrt{p_{22} - c_{21}^2}$
$p_{31} = c_{11}c_{31}$	$c_{31} = p_{31}/c_{11}$
$p_{32} = c_{21}c_{31} + c_{22}c_{32}$	$c_{32} = (p_{32} - c_{21}c_{31})/c_{22}$
$p_{33} = c_{31}^2 + c_{32}^2 + c_{33}^2$	$c_{33} = \sqrt{p_{33} - c_{31}^2 - c_{32}^2}$

A solution can also be obtained by stepping across the rows and then down the rows, in the order c_{11}, c_{21}, c_{31}, c_{22}, c_{32}, c_{33}.

The general solutions can be put in the form of algorithms looping through the rows and columns of C and using the prior results. The example above suggests two algorithmic solutions, one looping in row–column order and one looping in column–row order. There is also the choice of whether the solution C should be lower triangular or upper triangular.

Algorithmic solutions are given in Table 6.3. The one on the left can be implemented as `C=chol(M)'`, using the built-in MATLAB function `chol`. The one in the right column is implemented in the m-file `chol2.m`.

PROGRAMMING NOTE MATLAB automatically assigns the value zero to all the unassigned matrix locations. This will not be necessary if subsequent processes treat the resulting Cholesky factor matrix C as triangular and do not bother to add or multiply the zero elements.

6.4.3.2 Modified Cholesky (UD) Decomposition Algorithms

Unit Triangular Matrices An upper triangular matrix U is called *unit upper triangular* if its diagonal elements are all 1 (unity). Similarly, a lower triangular matrix L is called *unit lower triangular* if all of its diagonal elements are unity.

UD *Decomposition Algorithm* The *modified* Cholesky decomposition of a symmetric positive-definite matrix M is a decomposition into products $M = UDU^{\mathrm{T}}$ such that U is unit upper triangular and D is diagonal. It is also called *UD decomposition*.

A procedure for implementing *UD* decomposition is presented in Table 6.4. This algorithm is implemented in the m-file modchol.m. It takes M as input and

TABLE 6.3 Cholesky Decomposition Algorithms

Given an $m \times m$ symmetric positive-definite matrix M, a triangular matrix C such that $M = CC^{\mathrm{T}}$ is computed.

Lower Triangular Result

```
for j = 1: m,
  for i = 1 : j,
   sigma = M (i, j);
   for k = 1 : j - 1,
    sigma = sigma - C (i, k) *C (j, k);
   end;
   if i == j
    C (i, j) = sqrt (sigma);
   else
    C (i, j) = sigma/C (j, j)
   end;
  end;
 end;
```

Upper Triangular Result

```
for j = m: -1:1,
   for i = j : -1:1,
    sigma = M (i, j);
    for k = j + 1 : m,
     sigma = sigma - C (i, k) *C (j, k);
    end;
    if i == j
     C (i, j) = sqrt (sigma);
    else
    C (i, j) = sigma/C (j, j)
    end;
  end;
 end;
```

Computational complexity: $\frac{1}{6}m(m-1)(m+4)$ flops $+ m\sqrt{}$.

TABLE 6.4 *UD* Decomposition Algorithm

Given M, a symmetric, positive-definite $m \times m$ matrix, U and D, modified Cholesky factors of M, are computed, such that U is a unit upper triangular matrix, D is a diagonal matrix, and $M = UDU^{\mathrm{T}}$

```
for j = m : -1:1,
  for i = j : -1:1,
    sigma = M (i, j);
    for k = j + 1 : m,
     sigma = sigma - U (i, k) *D (k, k) *U (j, k);
    end;
    if i == j
     D (j, j) = sigma;
     U (j, j) = 1;
    else
     U (i, j) = sigma/D (j, j);
    end;
  end;
end;
```

Computational complexity: $\frac{1}{6}m(m-1)(m+4)$ flops.

returns U and D as output. The decomposition can also be implemented in place, overwriting the input array containing M with D (on the diagonal of the array containing M) and U (in the strictly upper triangular part of the array containing M). This algorithm is only slightly different from the upper triangular Cholesky decomposition algorithm presented in Table 6.3. The big difference is that the modified Cholesky decomposition does not require taking square roots.

6.4.3.3 Decorrelating Measurement Noise The decomposition methods developed for factoring the covariance matrix of estimation uncertainty may also be applied to the covariance matrix of measurement uncertainty, R. This operation redefines the measurement vector (via a linear transform of its components) such that its measurement errors are uncorrelated from component to component. That is, the new covariance matrix of measurement uncertainty is a *diagonal* matrix. In that case, the components of the redefined measurement vector can be processed serially as uncorrelated scalar measurements. The reduction in the computational complexity[4] of the Kalman filter with this approach will be covered in Section 6.6.1.

Suppose, for example, that

$$z = Hx + v \tag{6.15}$$

[4]The methodology used for determining the computational complexities of algorithms in this chapter is presented in Section 6.4.3.6.

is an observation with measurement sensitivity matrix H and noise v that is correlated from component to component of v. That is, the covariance matrix

$$E\langle vv^{\mathrm{T}}\rangle = R \tag{6.16}$$

is not a diagonal matrix. Then the scalar components of z cannot be processed serially as scalar observations with statistically independent measurement errors.

However, R can always be factored in the form

$$R = UDU^{\mathrm{T}}, \tag{6.17}$$

where D is a diagonal matrix and U is an upper triangular matrix. Unit triangular matrices have some useful properties:

- The determinant of a unit triangular matrix is 1. Unit triangular matrices are, therefore, always *nonsingular*. In particular, they always have a matrix inverse.
- The inverse of a unit triangular matrix is a unit triangular matrix. The inverse of a unit upper triangular matrix is unit upper triangular, and the inverse of a unit lower triangular matrix is a unit lower triangular matrix.

It is not necessary to compute U^{-1} to perform measurement decorrelation, but it is useful for pedagogical purposes to use U^{-1} to redefine the measurement as

$$\begin{aligned}
\acute{z} &= U^{-1}z && (6.18)\\
&= U^{-1}(Hx + v) && (6.19)\\
&= (U^{-1}H)x + (U^{-1}v) && (6.20)\\
&= \acute{H}x + \acute{v}. && (6.21)
\end{aligned}$$

That is, this "new" measurement $\acute{z}$ has measurement sensitivity matrix $\acute{H} = U^{-1}H$ and observation error $\acute{v} = U^{-1}v$. The covariance matrix R' of the observation error $\acute{v}$ will be the expected value

$$\begin{aligned}
R' &= E\langle \acute{v}\acute{v}^{\mathrm{T}}\rangle && (6.22)\\
&= E\langle (U^{-1}v)(U^{-1}v)^{\mathrm{T}}\rangle && (6.23)\\
&= E\langle U^{-1}vv^{\mathrm{T}}U^{T-1}\rangle && (6.24)\\
&= U^{-1}E\langle vv^{\mathrm{T}}\rangle U^{\mathrm{T}-1} && (6.25)\\
&= U^{-1}RU^{\mathrm{T}-1} && (6.26)\\
&= U^{-1}(UDU^{\mathrm{T}})U^{\mathrm{T}-1} && (6.27)\\
&= D. && (6.28)
\end{aligned}$$

That is, this redefined measurement has uncorrelated components of its measurement errors, which is what was needed for serializing the processing of the components of the new vector-valued measurement.

In order to decorrelate the measurement errors, one must solve the unit upper triangular system of equations

$$U\acute{z} = z \tag{6.29}$$

$$U\acute{H} = H \tag{6.30}$$

for $\acute{z}$ and $\acute{H}$, given z, H, and U. As noted previously, *it is not necessary to invert U* to solve for $\acute{z}$ and $\acute{H}$.

Solving Unit Triangular Systems It was mentioned above that it is not necessary to invert U to decorrelate measurement errors. In fact, it is only necessary to solve equations of the form $UX = Y$, where U is a unit triangular matrix and X and Y have conformable dimensions. The objective is to solve for X, given Y. It can be done by what is called *back substitution*. The algorithms listed in Table 6.5 perform the solutions by back substitution. The one on the right overwrites Y with $U^{-1}Y$. This feature is useful when several procedures are composed into one special-purpose procedure, such as the decorrelation of vector-valued measurements.

Specialization for Measurement Decorrelation A complete procedure for measurement decorrelation is listed in Table 6.6. It performs the UD decomposition and upper triangular system solution in place (overwriting H with $U^{-1}H$ and z with $U^{-1}z$) after decomposing R as $R = UDU^{\mathrm{T}}$ in place (overwriting the diagonal of R with $\acute{R} = D$ and overwriting the strictly upper triangular part of R with the strictly upper triangular part of U^{-1}).

TABLE 6.5 Unit Upper Triangular System Solution

Input: U, $m \times m$ unit upper triangular matrix; Y, $m \times p$ matrix Output: $X := U^{-1}Y$	Input: U, $m \times m$ unit upper triangular matrix; Y, $m \times p$ matrix Output: $Y := U^{-1}Y$

`for j = 1 : p,`
`  for i = m: -1:1,`
`  X (i, j) = Y (i, j);`
`  for k = i + 1 : m,`
`  X (i, j) = X (i, j) - U`
`  (i, k) *X (k, j);`
`  end;`
`  end;`
`end;`
	`for j = 1 : p,`
	`  for i = m: -1:1,`
	`  for k = i + 1 : m,`
	`  Y (i, j) = Y (i, j) - U`
	`  (i, k) *Y (k, j);`
	`  end;`
	`  end;`
	`end;`

Computational complexity: $pm(m-1)/2$ flops.	

TABLE 6.6 Measurement Decorrelation Procedure

The vector-valued measurement $z = Hx + v$, with correlated components of the measurement error $E(vv^{\mathrm{T}}) = R$, is transformed into the measurement $\acute{z} = \acute{H}x + \acute{v}$ with uncorrelated components of the measurement error $\acute{v}[E(\acute{v}\acute{v}\mathrm{T}) = \mathrm{D}$, a diagonal matrix], by overwriting H with $\acute{H} = U^{-1}H$ and z with $\acute{z} = U^{-1}z$, after decomposing R to UDU^{T}, overwriting the diagonal of R with D.

Symbol	Definition
R	Input: $\ell \times \ell$ covariance matrix of measurement uncertainty Output: D (on diagonal), U (above diagonal)
H	Input: $\ell \times n$ measurement sensitivity matrix Output: overwritten with $\acute{H} = U^{-1}H$
z	Input: measurement ℓ-vector Output: overwritten with $\acute{z} = U^{-1}z$

Procedure:
1. Perform UD decomposition of R in place (Table 6.4).
2. Solve $U\acute{z} = z$ and $U\acute{H} = H$ in place (Table 6.5).

Computational complexity: $\frac{1}{6}\ell(\ell-1)(\ell+4) + \frac{1}{2}\ell(\ell-1)(n+1)$ flops.

6.4.3.4 Symmetric Positive-Definite System Solution Cholesky decomposition provides an efficient and numerically stable method for solving equations of the form $AX = Y$ when A is a symmetric, positive-definite matrix. The modified Cholesky decomposition is even better, because it avoids taking scalar square roots. It is the recommended method for forming the term $[HPH^{\mathrm{T}} + R]^{-1}H$ in the conventional Kalman filter without explicitly inverting a matrix. That is, if one decomposes $HPH^{\mathrm{T}} + R$ as UDU^{T}, then

$$[UDU^{\mathrm{T}}][HPH^{\mathrm{T}} + R]^{-1}H = H. \tag{6.31}$$

It then suffices to solve

$$UDU^{\mathrm{T}}X = H \tag{6.32}$$

for X. This can be done by solving the three problems

$$UX_{[1]} = H \text{ for } X_{[1]}, \tag{6.33}$$

$$DX_{[2]} = X_{[1]} \text{ for } X_{[2]}, \tag{6.34}$$

$$U^{\mathrm{T}}X = X_{[2]} \text{ for } X. \tag{6.35}$$

The first of these is a unit upper triangular system, which was solved in the previous subsection. The second is a system of independent scalar equations, which has a simple solution. The last is a unit lower triangular system, which can be solved by *forward substitution*—a simple modification of back substitution. The computational complexity of this method is m^2p, where m is the row and column dimension of A and p is the column dimension of X and Y.

6.4.3.5 Transforming Covariance Matrices into Information Matrices The information matrix is the inverse of the covariance matrix—and vice versa. Although matrix inversion is generally to be avoided if possible, it is not possible to avoid it forever. This is one of those problems that require it.

The inversion is not possible unless one of the matrices (either P or Y) is positive-definite, in which case both will be positive-definite and they will have the same condition number. If they are sufficiently well conditioned, they can be inverted in place by UD decomposition, followed by inversion and recomposition in place. The in-place UD decomposition procedure is listed in Table 6.4. A procedure for inverting the result in place is shown in Table 6.7. A matrix inversion procedure using these two is outlined in Table 6.8. It should be used with caution, however.

6.4.3.6 Computational Complexities Using the general methods outlined in [86] and [99], one can derive the complexity formulas shown in Table 6.9 for methods using Cholesky factors.

6.4.4 Kalman Implementation with Decorrelation

It was pointed out by Kaminski [147] that the computational efficiency of the conventional Kalman observational update implementation can be improved by processing the components of vector-valued observations sequentially using the error decorrelation algorithm in Table 6.6, if necessary. The computational savings with the measurement decorrelation approach can be evaluated by comparing the rough operations counts of the two approaches using the operations counts for the sequential approach given in Table 6.10. One must multiply by ℓ, the number of operations required for the implementation of the scalar observational update equations, and add the number of operations required for performing the decorrelation.

The computational advantage of the decorrelation approach is

$$\frac{1}{3}\ell^3 - \frac{1}{2}\ell^2 + \frac{7}{6}\ell - \ell n + 2\ell^2 n + \ell n^2 \text{ flops.}$$

That is, it requires that many fewer flops to decorrelate vector-valued measurements and process the components serially.

TABLE 6.7 Unit Upper Triangular Matrix Inversion

Input/output: U, an $m \times m$ unit upper triangular matrix (U is overwritten with U^{-1})

```
for i = m : -1:1,
 for j = m : -1 : i + 1,
  U (i, j) = -U (i, j);
  for k = i + 1 : j - 1,
    U (i, j) = U (i, j) - U (i, k)*U (k, j);
  end;
 end;
end;
```

Computational complexity: $m(m-1)(m-2)/6$ flops.

TABLE 6.8 Symmetric Positive-Definite Matrix Inversion Procedure

Inverts a symmetric positive definite matrix in place

Symbol	Description
M	Input: $m \times m$ symmetric positive definite matrix Output: M is overwritten with M^{-1}

Procedure: 1. Perform UD decomposition of M in place.
2. Invert U in place (in the M-array).
3. Invert D in place: for $i = 1$: m, $M(i, i) = 1/M(i, i)$; end;
4. Recompose $M^{-1} = (U^{-\mathrm{T}}D^{-1})U^{-1}$ in place:

```
for j = m: -1:1,
   for i = j : -1 : 1,
    if i == j
       s = M(i,i);
    else
       s = M(i, i)*M(i, j);
    end;
    for k = 1 : i - 1,
      s = s + M(k,i)*M(k, k)*M(k, j);
    end;
    M(i, j) = s;
    M(j, i) = s;
  end;
end;
```

Computational complexity:	
UD decomp.	$m(m-1)(m+4)/6$
UD inverse	$m(m-1)(m-2)/6$
Recompose M^{-1}	$m(m-1)(2m+5)/6$
Total:	$m(m-1)(4m+7)/6$ flops

6.4.5 Symmetric Square Roots of Elementary Matrices

Elementary Matrices An elementary matrix is a matrix of the form $I - s\mathbf{v}\mathbf{w}^{\mathrm{T}}$, where I is an identity matrix, s is a scalar, and $\mathbf{v}$, $\mathbf{w}$ are column vectors of the same row dimension as I. Elementary matrices have the property that their products are also elementary matrices. Their squares are also elementary matrices, with the same vector values ($\mathbf{v}$, $\mathbf{w}$) but with different scalar values (s).

Symmetric Elementary Matrices An elementary matrix is symmetric if $\mathbf{v} = \mathbf{w}$. The squares of such matrices have the same format:

$$(I - \sigma\mathbf{v}\mathbf{v}^{\mathrm{T}})^2 = (I - \sigma\mathbf{v}\mathbf{v}^{\mathrm{T}})(I - \sigma\mathbf{v}\mathbf{v}^{\mathrm{T}}) \tag{6.36}$$

$$= I - 2\sigma\mathbf{v}\mathbf{v}^{\mathrm{T}} + \sigma^2|\mathbf{v}|^2\mathbf{v}\mathbf{v}^{\mathrm{T}} \tag{6.37}$$

$$= I - (2\sigma - \sigma^2|\mathbf{v}|^2)\mathbf{v}\mathbf{v}^{\mathrm{T}} \tag{6.38}$$

$$= I - s\,\mathbf{v}\mathbf{v}^{\mathrm{T}} \tag{6.39}$$

$$s = (2\sigma - \sigma^2|\mathbf{v}|^2). \tag{6.40}$$

TABLE 6.9 Computational Complexity Formulas

Cholesky decomposition of an $m \times m$ matrix:

$$\mathcal{C}_{\text{Cholesky}} = \sum_{j=1}^{m}\left[m - j + \sum_{i=j}^{m}(m-j)\right]$$
$$= \frac{1}{3}m^3 + \frac{1}{2}m^2 - \frac{5}{6}m$$

UD decomposition of an $m \times m$ matrix:

$$\mathcal{C}_{UD} = \sum_{j=1}^{m}\left[m - j + \sum_{i=j}^{m}2(m-j)\right]$$
$$= \frac{2}{3}m^3 + \frac{1}{2}m^2 - \frac{7}{6}m$$

Inversion of an $m \times m$ unit triangular matrix:

$$\mathcal{C}_{\text{UTINV}} = \sum_{i=1}^{m-1}\sum_{j=i+1}^{m}(j - i - 1)$$
$$= \frac{1}{6}m^3 - \frac{1}{2}m^2 + \frac{1}{3}m$$

Measurement decorrelation ($\ell \times n$ H-matrix):

$$\mathcal{C}_{\text{DeCorr}} = \mathcal{C}_{UD} + \sum_{i=1}^{\ell-1}\sum_{k=i+1}^{\ell}(n+1)$$
$$= \frac{2}{3}\ell^3 + \ell^2 - \frac{5}{3}\ell + \frac{1}{2}\ell^2 n - \frac{1}{2}\ell n$$

Inversion of an $m \times m$ covariance matrix:

$$\mathcal{C}_{\text{COVINV}} = \mathcal{C}_{UD} + \mathcal{C}_{\text{UTINV}} + m + \sum_{i=1}^{m}[i(i-1)(m-i+1)]$$
$$= m^3 + \frac{1}{2}m^2 + \frac{1}{2}m$$

TABLE 6.10 Operations for Sequential Processing of Measurements

Operation	Flops
$H \times P(-)$	n^2
$H \times [HP(-)]^{\mathrm{T}} + R$	n
$\{H[HP(-)]^{\mathrm{T}} + R\}^{-1}$	1
$\{H[HP(-)]^{\mathrm{T}} + R\}^{-1} \times [HP(-)]$	n
$P(-) - [HP(-)] \times \{H[HP(-)]^{\mathrm{T}} + R\}^{-1}[HP(-)]$	$\frac{1}{2}n^2 + \frac{1}{2}n$
Total (per component) $\times$ ℓ components	$\left(\frac{1}{2}n^2 + \frac{5}{2}n + 1\right) \times \ell$
+ decorrelation complexity	$\frac{2}{3}\ell^3 + \ell^2 - \frac{5}{3}\ell + \frac{1}{2}\ell^2 n - \frac{1}{2}\ell n$
Total	$\frac{2}{3}\ell^3 + \ell^2 - \frac{2}{3}\ell + \frac{1}{2}\ell^2 n + 2\ell + n\frac{1}{2}\ell n^2$

Symmetric Square Root of a Symmetric Elementary Matrix One can also invert the last equation above and take the square root of the symmetric elementary matrix $(I - s\mathbf{v}\mathbf{v}^{\mathrm{T}})$. This is done by solving the scalar quadratic equation

$$s = 2\sigma - \sigma^2|\mathbf{v}|^2, \tag{6.41}$$

$$\sigma^2|\mathbf{v}|^2 - 2\sigma + s = 0 \tag{6.42}$$

to obtain the solution

$$(I - s\,\mathbf{v}\mathbf{v}^{\mathrm{T}})^{1/2} = I - \sigma\mathbf{v}\mathbf{v}^{\mathrm{T}}, \tag{6.43}$$

$$\sigma = \frac{1 + \sqrt{1 - s|\mathbf{v}|^2}}{|\mathbf{v}|^2}. \tag{6.44}$$

In order that this square root be a real matrix, it is necessary that the radicand

$$1 - s|\mathbf{v}|^2 \geq 0. \tag{6.45}$$

6.4.6 Triangularization Methods

6.4.6.1 Triangularization Methods for Least-Squares Problems These techniques were originally developed for solving least-squares problems. The overdetermined system

$$Ax = b$$

can be solved efficiently and relatively accurately by finding an orthogonal matrix T such that the product $B = TA$ is a triangular matrix. In that case, the solution to the triangular system of equations

$$Bx = Tb$$

can be solved by backward substitution.

6.4.6.2 Triangularization (QR Decomposition) of **A** It is a theorem of linear algebra that any general matrix A can be represented as a product[5]

$$A = C_{k+1}(-)T \tag{6.46}$$

[5]This is the so-called QR decomposition in disguise. It is customarily represented as $A = QR$ (whence the name), where Q is orthogonal and R is triangular. However, as mentioned earlier, we have already committed the symbols Q and R to play other roles in this book. In this instance of QR decomposition, it has the transposed form $A^{\mathrm{T}} = T^{\mathrm{T}}C^{\mathrm{T}}_{k+\,1}(-)$, where T^{T} is the stand-in for the original Q (the orthogonal factor) and $C^{\mathrm{T}}_{k+\,1}(-)$ is the stand-in for the original R (the triangular factor).

of a triangular matrix $C_{k+1}(-)$ and an orthogonal matrix T. This type of decomposition is called *QR decomposition* or *QR triangularization*. By means of this triangularization, the symmetric matrix product factorization

$$P_{k+1}(-) = AA^{\mathrm{T}} \tag{6.47}$$

$$= [C_{k+1}(-)T][C_{k+1}(-)T]^{\mathrm{T}} \tag{6.48}$$

$$= C_{k+1}(-)TT^{\mathrm{T}}C_{k+1}^{\mathrm{T}}(-) \tag{6.49}$$

$$= C_{k+1}(-)(TT^{\mathrm{T}})C_{k+1}^{\mathrm{T}}(-) \tag{6.50}$$

$$= C_{k+1}(-)C_{k+1}^{\mathrm{T}}(-) \tag{6.51}$$

also defines a triangular Cholesky decomposition of $C_{k+1}(-)$ of $P_{k+1}(-)$. This is the basis for performing temporal updates of Cholesky factors of P.

6.4.6.3 Uses of Triangularization in Kalman Filtering Matrix triangularization methods were originally developed for solving least-squares problems. They are used in Kalman filtering for

- temporal updates of Cholesky factors of the covariance matrix of estimation uncertainty, as described above;
- observational updates of Cholesky factors of the estimation information matrix, as described in Section 6.6.3.5; and
- combined updates (observational and temporal) of Cholesky factors of the covariance matrix of estimation uncertainty, as described in Section 6.6.2.

A modified Givens rotation due to Gentleman [96] is used for the temporal updating of modified Cholesky factors of the covariance matrix.

In these applications, as in most least-squares applications, the orthogonal matrix factor is unimportant. The resulting triangular factor is the intended result, and numerically stable methods have been developed for computing it.

6.4.6.4 Triangularization Algorithms Two of the more stable methods for matrix triangularization are presented in the following subsections. These methods are based on orthogonal transformations (matrices) that, when applied to (multiplied by) general matrices, reduce them to triangular form. Both were published in the same year (1958). Both define the requisite transformation as a product of "elementary" orthogonal transformations:

$$T = T_1 T_2 T_3 \cdots T_m. \tag{6.52}$$

These elementary transformations are either *Givens rotations* or *Householder reflections*. In each case, triangularization is achieved by zeroing of the nonzero elements

on one side of the main diagonal. Givens rotations zero these elements one by one. Householder reflections zero entire subrows of elements (i.e., the part of a row to the left of the triangularization diagonal) on each application. The order in which such transformations may be applied must be constrained so that they do not "unzero" previously zeroed elements.

6.4.6.5 Triangularization by Givens Rotations This method for triangularization, due to Givens [97], uses a *plane rotation matrix* $T_{ij}(\theta)$ of the following form:

$$T_{ij}(\theta) = \begin{array}{c} \\ \\ j \\ \\ \\ \\ \\ \\ i \\ \\ \\ \\ \end{array} \begin{array}{c} \begin{array}{ccccccccccc} & & j & & i & & & & & & \end{array} \\ \begin{pmatrix} 1 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ 0 & \cdots & 1 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & \cos(\theta) & 0 & \cdots & 0 & \sin(\theta) & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 0 & 1 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ \vdots & & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & & \vdots \\ 0 & \cdots & 0 & & 0 & \cdots & 1 & 0 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & -\sin(\theta) & 0 & \cdots & 0 & \cos(\theta) & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 1 & \cdots & 0 \\ \vdots & & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 1 \end{pmatrix} \end{array} \tag{6.53}$$

It is also called a *Givens rotation matrix* or *Givens transformation matrix*. Except for the *i*th and *j*th rows and columns, the plane rotation matrix has the appearance of an identity matrix. When it is multiplied on the right-hand side of another matrix, it affects only the *i*th and *j*th columns of the matrix product. It rotates the *i*th and *j*th elements of a row or column vector, as shown in Figure 6.3. It can be used to rotate one of the components all the way to zero, which is how it is used in triangularization.

Triangularization of a matrix A by Givens rotations is achieved by successive multiplications of A on one side by Givens rotation matrices, as illustrated by the following example.

Example 6.7 (Givens Trianglarization) Consider the problem of upper triangularizing the 2×3 symbolic matrix

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} \tag{6.54}$$

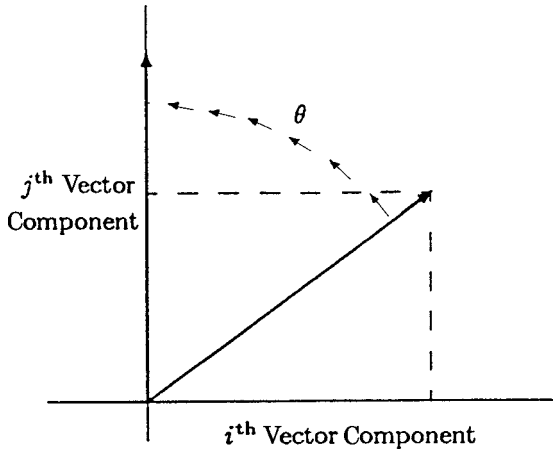


Figure 6.3 Component transformations by plane rotation.

by multiplying with Givens rotation matrices on the right. The first product

$$\begin{aligned}\acute{A}(\theta) &= AT_{23}(\theta)\\ &= \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & \sin(\theta) \\ 0 & -\sin(\theta) & \cos(\theta) \end{bmatrix}\\ &= \begin{bmatrix} a_{11} & a_{12}\cos(\theta) - a_{13}\sin(\theta) & a_{12}\sin(\theta) + a_{13}\cos(\theta) \\ a_{21} & \boxed{a_{22}\cos(\theta) - a_{23}\sin(\theta)} & a_{22}\sin(\theta) + a_{23}\cos(\theta) \end{bmatrix}.\end{aligned}$$

The framed element in the product will be zero if $a_{22}^2 + a_{23}^2 = 0$, and if $a_{22}^2 + a_{23}^2 > 0$, the values

$$\cos(\theta) = \frac{a_{23}}{\sqrt{a_{22}^2 + a_{23}^2}}, \quad \sin(\theta) = \frac{a_{22}}{\sqrt{a_{22}^2 + a_{23}^2}}.$$

will force it to zero. The resulting matrix $\acute{A}$ can be multiplied again on the right by the Givens rotation matrix $T_{13}(\acute{\theta})$ to yield yet a second intermediate matrix form

$$\begin{aligned}\breve{A}(\acute{\theta}) &= AT_{23}(\theta)T_{13}(\acute{\theta})\\ &= \begin{bmatrix} a_{11} & \acute{a}_{12} & \acute{a}_{13} \\ a_{21} & 0 & \acute{a}_{23} \end{bmatrix} \begin{bmatrix} \cos(\acute{\theta}) & & \sin(\acute{\theta}) \\ 0 & 1 & 0 \\ -\sin(\acute{\theta}) & 0 & \cos(\acute{\theta}) \end{bmatrix}\\ &= \begin{bmatrix} \acute{a}_{11} & \acute{a}_{12} & \acute{a}_{13} \\ 0 & 0 & \acute{a}_{23} \end{bmatrix}\end{aligned}$$

for $\acute{\theta}$ such that

$$\cos(\acute{\theta}) = \frac{\acute{a}_{23}}{\sqrt{\acute{a}_{21}^2 + \acute{a}_{23}^2}}, \quad \sin(\acute{\theta}) = \frac{\acute{a}_{21}}{\sqrt{\acute{a}_{21}^2 + \acute{a}_{23}^2}}.$$

A third Givens rotation yields the final matrix form

$$\begin{aligned}\breve{A}(\acute{\theta}) &= AT_{23}(\theta)T_{13}(\acute{\theta})T_{12}(\acute{\theta}) \\ &= \begin{bmatrix} \acute{a}_{11} & \acute{a}_{12} & \acute{a}_{13} \\ 0 & 0 & \acute{a}_{23} \end{bmatrix} \begin{bmatrix} \cos(\acute{\theta}) & \sin(\acute{\theta}) & 0 \\ -\sin(\acute{\theta}) & \cos(\acute{\theta}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 0 & \breve{a}_{12} & \acute{a}_{13} \\ 0 & 0 & \grave{a}_{23} \end{bmatrix}\end{aligned}$$

for $\acute{\theta}$ such that

$$\cos(\acute{\theta}) = \frac{\acute{a}_{12}}{\sqrt{\acute{a}_{11}^2 + \acute{a}_{12}^2}}, \quad \sin(\acute{\theta}) = \frac{\acute{a}_{11}}{\sqrt{\acute{a}_{11}^2 + \acute{a}_{12}^2}}.$$

The remaining nonzero part of this final result is an upper triangular submatrix right adjusted within the original array dimensions.

The order in which successive Givens rotation matrices are applied is constrained to avoid unzeroing elements of the matrix that have already been zeroed by previous Givens rotations. Figure 6.4 shows the constraints that guarantee such noninterference. If we suppose that the element to be annihilated (designated by x in the figure) is in the ith column and kth row and the corresponding diagonal element of the soon-to-be triangular matrix is in the jth column, then it is sufficient if the elements below the kth rows in those two columns have already been annihilated by Givens rotations. The reason for this is simple: the Givens rotations can only form linear combinations of row elements in those two columns. If those row elements are already zero, then any linear combination of them will also be zero. The result: no effect.

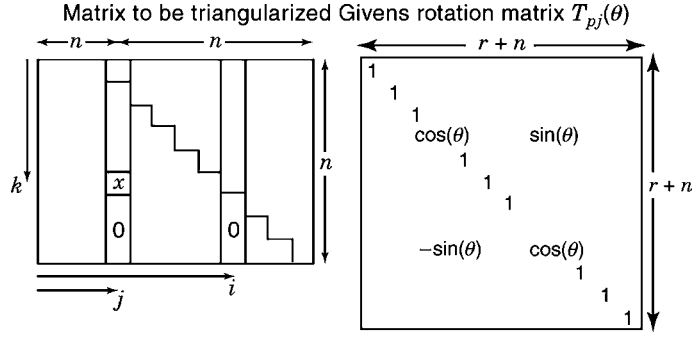


Figure 6.4 Constraints on Givens triangularization order.

A Givens Triangularization Algorithm The method used in the previous example can be generalized to an algorithm for upper triangularization of an $n \times (n + r)$ matrix, as follows:

```
Input: A, a n - by (n + r) matrix
Output: A is overwritten by an upper triangular matrix C,
        right-adjusted in the array, such that output
        value of CC' equals input value of AA'.

        for i = n: -1:1,
          for j = 1: r + i,
            rho = sqrt (A (i, r + i)^ 2+A (i, j)^2);
            s = A (i, j)/ rho;
            c = A (i, r + i)/ rho;
            for k = 1: i,
                          x = c*A (k, j) - s*A (k, r + i);
             A (k, r + i) = s*A (k, j) + c*A (k, r + i);
                A (k, j) = x;
            end;
          end;
        end;
```

In this particular form of the algorithm, the outermost loop (the loop with index i in the listing) zeros elements of A one row at a time. An analogous algorithm can be designed in which the outermost loop is by columns.

6.4.6.6 Triangularization by Householder Reflections This method of triangularization was discovered by Householder [112]. It uses an elementary matrix of the form

$$T(v) = I - \frac{2}{v^{\mathrm{T}}v} vv^{\mathrm{T}}, \tag{6.55}$$

where v is a column vector and I is the identity matrix of the same dimension. This particular form of the elementary matrix is called a *Householder reflection, Householder transformation*, or *Householder matrix*.

Note that Householder transformation matrices are always *symmetric*. They are also *orthogonal*, for

$$T(v)T^{\mathrm{T}}(v) = \left(I - \frac{2}{v^{\mathrm{T}}v} vv^{\mathrm{T}}\right)\left(I - \frac{2}{v^{\mathrm{T}}v} vv^{\mathrm{T}}\right) \tag{6.56}$$

$$= I - \frac{4}{v^{\mathrm{T}}v} vv^{\mathrm{T}} + \frac{4}{(v^{\mathrm{T}}v)^2} v(v^{\mathrm{T}}v)v^{\mathrm{T}} \tag{6.57}$$

$$= I. \tag{6.58}$$

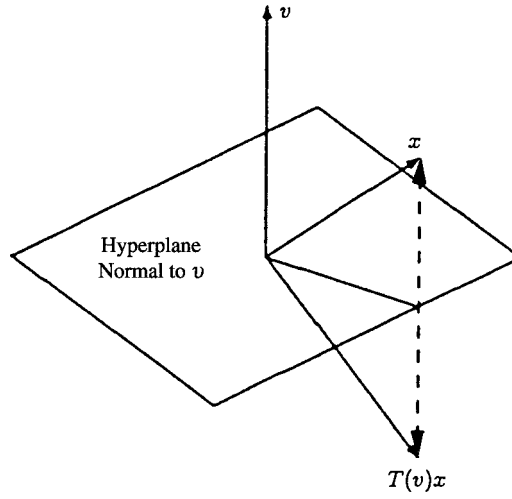


Figure 6.5 Householder reflection of a vector **x**.

They are called *reflections* because they transform any matrix x into its "mirror reflection" in the plane (or hyperplane[6]) normal to the vector v, as illustrated in Figure 6.5 (for three-dimensional v and x). By choosing the proper mirror plane, one can place the reflected vector $T(v)x$ along any direction whatsoever, including parallel to any coordinate axis.

Example 6.8 (Householder Reflection Along One Coordinate Axis) Let x be any n-dimensional row vector, and let

$$e_k \stackrel{\text{def}}{=} (0 \quad 0 \quad 0 \quad \cdots \quad 0 \quad \overset{k}{1} \quad 0 \quad \cdots \quad 0)$$

be the kth row of the $n \times n$ identity matrix. If the vector v of the Householder reflection $T(v)$ is defined as

$$v = x^{\mathrm{T}} + \alpha e_k^{\mathrm{T}},$$

where α is a scalar, then the inner products

$$v^{\mathrm{T}}v = |x|^2 + 2\alpha x_k + \alpha^2,$$

$$xv = |x|^2 + \alpha x_k,$$

[6]The dimension of the hyperplane normal to the vector v will be one less than the dimension of the space containing v. When, as in the illustration, v is a three-dimensional vector (i.e., the space containing v is three-dimensional), the hyperplane normal to v is a two-dimensional plane.

where x_k is the kth component of x. The Householder reflection $xT(v)$ of x will be

$$\begin{aligned} xT(v) &= x\left(I - \frac{2}{v^{\mathrm{T}}v}vv^{\mathrm{T}}\right) \\ &= x\left(I - \frac{2}{x^{\mathrm{T}}x + 2\alpha x_k + \alpha^2}vv^{\mathrm{T}}\right) \\ &= x - \frac{2xv}{|x|^2 + 2\alpha x_k + \alpha^2}v^{\mathrm{T}} \\ &= x - \frac{2(|x|^2 + \alpha x_k)}{|x|^2 + 2\alpha x_k + \alpha^2}(x + \alpha e_k) \\ &= \left[1 - \frac{2(|x|^2 + \alpha x_k)}{|x|^2 + 2\alpha x_k + \alpha^2}\right]x - \left[\frac{2\alpha(|x|^2 + \alpha x_k)}{|x|^2 + 2\alpha x_k + \alpha^2}\right]e_k \\ &= \left[\frac{\alpha^2 - |x|^2}{|x|^2 + 2\alpha x_k + \alpha^2}\right]x - \left[\frac{2\alpha(|x|^2 + \alpha x_k)}{|x|^2 + 2\alpha x_k + \alpha^2}\right]e_k. \end{aligned}$$

Consequently, if one lets

$$\alpha = \mp|x|\,, \tag{6.59}$$

then

$$xT(v) = \pm|x|e_k\,. \tag{6.60}$$

That is, $xT(v)$ is parallel to the kth coordinate axis.

If, in the above example, x were the last row vector of an $n \times (n + r)$ matrix

$$M = \begin{bmatrix} Z \\ x \end{bmatrix},$$

and letting $k = 1$, then

$$MT(v) = \begin{bmatrix} ZT(v) \\ xT(v) \end{bmatrix} \tag{6.61}$$

$$= \begin{bmatrix} & & & ZT(v) & & \\ 0 & 0 & 0 & \cdots & 0 & |x| \end{bmatrix}, \tag{6.62}$$

the first step in upper triangularizing a matrix by Householder reflections.

Upper Triangularization by Successive Householder Reflections A single Householder reflection can be used to zero all the elements to the left of the diagonal in an entire row of a matrix, as shown in Figure 6.6. In this case, the vector x to be operated upon by the Householder reflection is a row vector composed of the first k components of a row of a matrix. Consequently, the dimension of the Householder

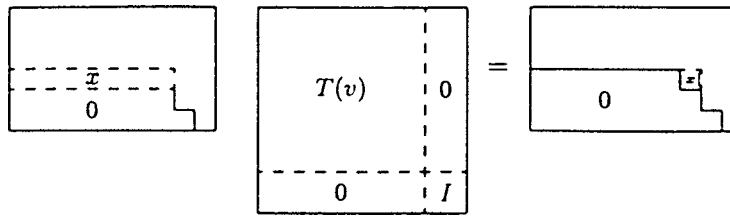


Figure 6.6 Zeroing one subrow with a Householder reflection.

reflection matrix $T(v)$ need only be k, which may be strictly less than the number of columns in the matrix to be triangularized. The "undersized" Householder matrix is placed in the upper left corner of the transformation matrix, and the remaining diagonal block is filled with an (appropriately dimensioned) identity matrix I, such that the row dimension of the resulting transformation matrix equals the column dimension of the matrix to be triangularized. The resulting composite matrix will always be orthogonal, so long as $T(v)$ is orthogonal.

There are two important features of this construction. The first is that the presence of the identity matrix has the effect of leaving the columns to the right of the kth column undisturbed by the transformation. The second is that the transformation does not unzero previously zeroed rows below. Together, these features allow the matrix to be triangularized by the sequence of transformations shown in Figure 6.7. Note that the *size* of the Householder transformation shrinks with each step.

Householder Triangularization Algorithm The algorithm listed below performs upper triangularization of a rectangular matrix $n \times (n + r)$ A in place using a scratch-pad $(n + r)$-vector v. The result is an upper triangular $n \times n$ matrix, right adjusted in the array A.

This algorithm includes a rescaling computation (involving the absolute value function `abs`) for better numerical stability. It is modified after the one given by

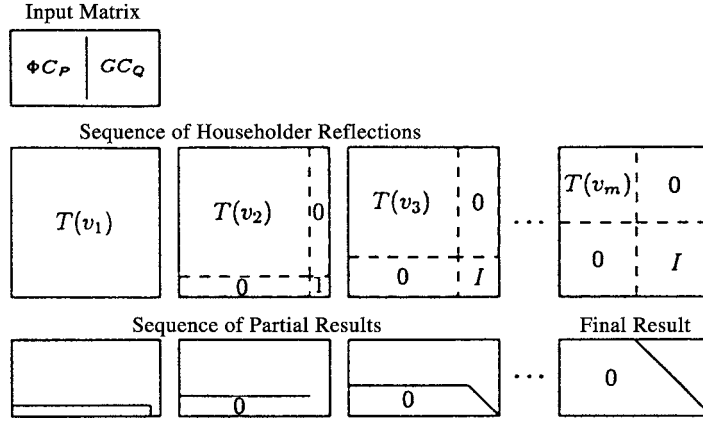


Figure 6.7 Upper triangularization by Householder reflections.

Golub and Van Loan [99] for Householder triangularization. The modification is required for applying the Householder transformations from the right, rather than from the left, and for the particular form of the input matrix used in the Kalman filter implementation. Further specialization of this algorithm for Kalman filtering is presented later in Table 6.14.

```
for k = n: -1:1,
    sigma = 0;
    for j = 1: r + k,
       sigma = sigma + A (k, j)^ 2;
    end;
    a = sqrt (sigma);
    sigma = 0;
    for j = 1: r + k,
       if j == r + k
         v (j) = A (k, j) - a;
       else
         v (j) = A (k, j);
       end;
       sigma = sigma + v (j)^ 2;
    end;
    a = 2/sigma;
    for i = 1 : k,
    sigma = 0;
    for j = 1 : r + k,
       sigma = sigma + A (i, j)*v (j);
    end;
    b = a* sigma;
    for j = 1 : r + k,
        A (i, j) = A (i, j) - b* v (j);
    end;
end;
```

6.5 SQUARE-ROOT AND *UD* FILTERS

Square-root filtering uses a reformulation of the Riccati equations such that the dependent variable is a Cholesky factor (or modified Cholesky factor) of the state estimation uncertainty matrix P. We present here just two of the many forms of square-root Kalman filtering, with other forms presented in the following section. The two selected forms of square-root filtering are

1. Carlson–Schmidt square-root filtering, which uses Cholesky factors of P, and
2. Bierman–Thornton *UD* filtering, which uses modified Cholesky factors of P.

These are perhaps the more favored implementation forms (after the conventional Kalman filter), because they have been used extensively and successfully on many problems that would be too poorly conditioned for conventional Kalman filter implementation. The *UD* filter, in particular, has been used successfully on problems with thousands of state variables.

This does not necessarily imply that these two methods are to be preferred above all others, however. It may be possible that the Morf–Kailath combined square-root filter (Section 6.6.2), for example, performs as well or better, but we are currently not aware of any comparable experience using that method.

6.5.1 Carlson–Schmidt Square-Root Filtering

This is a matched pair of algorithms for the observational and temporal update of the covariance matrix P in terms of its Cholesky factors. If the covariance matrices R (measurement noise) and Q (dynamic process noise) are not diagonal matrices, the implementation also requires *UD* or Cholesky factorization of these matrices.

6.5.1.1 Carlson "Fast Triangular" Update This algorithm is implemented in the MATLAB m-file `carlson.m` on the accompanying diskette. The algorithm is due to Carlson [57]. It generates an upper triangular Cholesky factor W for the Potter factorization and has generally lower computational complexity than the Potter algorithm. It is a specialized and simplified form of an algorithm used by Agee and Turner [2] for Kalman filtering. It is a rank 1 modification algorithm, like the Potter algorithm, but it produces a triangular Cholesky factor. It can be derived from Problem 6.14.

In the case that $m = j$, the summation on the left-hand side of Equation 6.166 has but one term:

$$W_{ij}W_{ij} = \Delta_{ij} - \frac{\mathbf{v}_i\mathbf{v}_j}{R + \sum_{k=1}^{j}\mathbf{v}_k^2}, \tag{6.63}$$

which can be solved in terms of the elements of the upper triangular Cholesky factor W:

$$W_{ij} = \begin{cases} 0, & i > j, \\ \sqrt{\dfrac{R + \sum_{k=1}^{j-1}\mathbf{v}_k^2}{R + \sum_{k=1}^{j}\mathbf{v}_k^2}}, & i = j, \\ \dfrac{-\mathbf{v}_i\mathbf{v}_j}{\left(R + \sum_{k=1}^{j-1}\mathbf{v}_k^2\right)\left(R + \sum_{k=1}^{j}\mathbf{v}_k^2\right)}, & i < j. \end{cases} \tag{6.64}$$

Given the above formula for the elements of W, one can derive the formula for the elements of $C(+) = C(-)W$, the upper triangular Cholesky factor of the *a posteriori*

covariance matrix of estimation uncertainty $P(+)$, given $C(-)$, the upper triangular Cholesky factor of the *a priori* covariance matrix $P(-)$.

Because both C and W are upper triangular, the elements $C_{ik} = 0$ for $k < i$ and the elements $W_{kj} = 0$ for $k > j$. Consequently, for $1 \le i \le j \le n$, the element in the ith row and jth column of the matrix product $C(-)W = C(+)$ will be

$$C_{ij}(+) = \sum_{k=i}^{j} C_{ik}(-)W_{kj} + \text{terms with zero factors} \tag{6.65}$$

$$= C_{ij}(-)W_{jj} + \sum_{k=i}^{j-1} C_{ik}(-)W_{kj} \tag{6.66}$$

$$= C_{ij}(-)\sqrt{\frac{R + \sum_{k=1}^{j-1} \mathbf{v}_k^2}{R + \sum_{k=1}^{j} \mathbf{v}_k^2}} - \sum_{k=i}^{j-1} \frac{C_{ik}(-)\mathbf{v}_k \mathbf{v}_j}{\left(R + \sum_{k=1}^{j-1} \mathbf{v}_k^2\right)\left(R + \sum_{k=1}^{j} \mathbf{v}_k^2\right)} \tag{6.67}$$

$$= \left(R + \sum_{k=1}^{j} \mathbf{v}_k^2\right)^{-1/2} \times \left[C_{ij}(-)\sqrt{R + \sum_{k=1}^{j-1} \mathbf{v}_k^2} - \frac{\mathbf{v}_j \sum_{k=i}^{j-1} C_{ik}(-)\mathbf{v}_k}{(R + \sum_{k=1}^{j-1} \mathbf{v}_k^2)^{1/2}}\right]. \tag{6.68}$$

This is a general formula for the upper triangular *a posteriori* Cholesky factor of the covariance matrix of estimation uncertainty, in terms of the upper triangular *a priori* Cholesky factor $C(-)$ and the vector $\mathbf{v} = C^{\mathrm{T}} H^{\mathrm{T}}$, where H is the measurement sensitivity matrix (a row vector). An algorithm implementing the formula is given in Table 6.11. This algorithm performs the complete observational update, including the update of the state estimate, in place. [Note that this algorithm forms the product $\mathbf{v} = C^{\mathrm{T}}(-)H^{\mathrm{T}}$ internally, computing and using the components $\sigma = \mathbf{v}_j$ as needed without storing the vector $\mathbf{v}$. It does store and use the vector $\mathbf{w} = C(-)\mathbf{v}$, the unscaled Kalman gain, however.]

It is possible—and often desirable—to save array space by storing triangular matrices in singly subscripted arrays. An algorithm (in FORTRAN) for such an implementation of this algorithm is given in Carlson's original paper [57].

6.5.1.2 Schmidt Temporal Update

Nonsquare, Nontriangular Cholesky Factor of $\mathrm{P_k}(-)$ If C_P is a Cholesky factor of $P_{k-1}(+)$ and C_Q is a Cholesky factor of Q_k, then the partitioned $n \times (n + q)$ matrix

$$A = [G_k C_Q \quad | \quad \Phi_k C_P] \tag{6.69}$$

TABLE 6.11 Carlson's Fast Triangular Observational Update

Symbol	Definition
z	Value of scalar measurement
R	Variance of scalar measurement uncertainty
H	Scalar measurement sensitivity vector ($1 \times n$ matrix)
C	Cholesky factors of $P(-)$ (input) and $P(+)$ (output)
x	State estimates $x(-)$ (input) and $x(+)$ (output)
w	Unscaled Kalman gain (internal)

```
alpha = R;
delta = z;
for j = 1: n,
   delta = delta - H (j)*x (j);
   sigma = 0;
   for i = 1: j,
      sigma = sigma + C (i, j)*H (i);
   end;
   beta = alpha;
   alpha = alpha + sigma^2;
   gamma = sqrt (alpha*beta);
   eta = beta/gamma;
   zeta = sigma/gamma;
   w (j) = 0;
   for i = 1: j,
      tau = C (i, j);
      C (i, j) = eta*C (i, j) - zeta*w (i);
      w (i) = w (i) + tau*sigma;
   end;
end;
epsilon = delta/alpha;
for i = 1: n,
     x (i) = x (i) + w (i)*epsilon;
end;
```

Computational complexity: $(2n^2 + 7n + 1)$ flops + $n\sqrt{}$.

has the $n \times n$ symmetric matrix product value

$$AA^{\mathrm{T}} = [G_{k-1}C_Q|\Phi_{k-1}C_P][G_{k-1}C_Q|\Phi_{k-1}C_P]^{\mathrm{T}} \tag{6.70}$$

$$= \Phi_{k-1}C_P C_P^{\mathrm{T}}\Phi_{k-1}^{\mathrm{T}} + G_{k-1}C_Q C_Q^{\mathrm{T}}G_{k-1}^{\mathrm{T}} \tag{6.71}$$

$$= \Phi_{k-1}P_{k-1}(+)\Phi_{k-1}^{\mathrm{T}} + G_{k-1}Q_{k-1}G_{k-1}^{\mathrm{T}} \tag{6.72}$$

$$= P_k(-) \tag{6.73}$$

That is, A is a nonsquare, nontriangular Cholesky factor of $P_k(-)$. If *only* it were square and triangular, it would be the kind of Cholesky factor we are looking for. It is not, but fortunately, there are algorithmic procedures for modifying A to that format.

PROGRAMMING NOTE Computation of GC_Q and ΦC_P *in place. This should be attempted only if memory limitations demand it*. The product ΦC_P can be computed in place by overwriting Φ with the product ΦC_P. This is not desirable if Φ is constant, however. (It is possible to overwrite C_P with the product ΦC_P, but this requires the use of an additional n-vector of storage. This option is left as an exercise for the truly needy.) Similarly, the product GC_Q can be computed in place by overwriting G with the product GC_Q if $r \leq n$. Algorithms for doing the easier in-place operations when the Cholesky factors C_Q and C_P are *upper* triangular are shown in Table 6.12. Note that these have roughly half the computational complexities of the general matrix products.

Triangularization Using Givens Rotations The Givens triangularization algorithm is presented in Section 6.4.6 The specialization of this algorithm to use GC_Q and ΦC_P in place, without having to stuff them into a common array, is shown in Table 6.13.

The computational complexity of Givens triangularization is greater than that of Householder triangularization, which is covered next. There are two attributes of the Givens method that might recommend it for certain applications, however:

1. The Givens triangularization procedure can exploit sparseness of the matrices to be triangularized. Because it zeros elements one at a time, it can skip over elements that are already zero. (The procedure may unzero some elements that are already zero in the process, however.) This will tend to decrease the computational complexity of the application.
2. The Givens method can be "parallelized" to exploit concurrent processing capabilities if they are available. The parallelized Givens method has no data contention among concurrent processes. They are working with data in different parts of the matrix as they are being triangularized.

Schmidt Temporal Updates Using Householder Reflections The basic Householder triangularization algorithm (see Section 6.4.6.2) operates on a single

TABLE 6.12 Algorithms Performing Matrix Operations in Place

Overwriting Φ by ΦC_P	Overwriting G by GC_Q

`for j = n : -1:1,`
`  for i = 1 : n,`
`  sigma = 0;`
`  for k = 1 : j,`
`  sigma = sigma + Phi (i, k)*CP (k, j);`
`  end;`
`  Phi (i, j) = sigma;`
`  end;`
`end;`
	`for j = r : -1:1,`
	`  for i = 1 : n,`
	`  sigma = 0;`
	`  for k = 1 : j,`
	`  sigma = sigma + G (i, k)*CQ (kj);`
	`  end;`
	`  G (i, j) = sigma;`
	`  end;`
	`end;`

Computational complexities	
$n^2(n+1)/2$	$nr(r+1)/2$

TABLE 6.13 Temporal Update by Givens Rotations

Symbol	Description
A	Input: $G_k\, C_{Q_k}$, an $n \times r$ matrix. Output: A is overwritten by intermediate results.
B	Input: $\Phi_k C_{P_k}(+)$, an $n \times n$ matrix. Output: B is overwritten by the upper triangular matrix $C_{P_{k+1}}(-)$.

```
for i = n: -1:1,
  for j = 1: r,
    rho = sqrt (B (i, i)^2 + A (i, j)^2);
    s = A (i, j)/rho;
    C = B (i, i)/rho;
    for k = 1: i,
      tau = c*A (k, j) - s*B (k, i);
      B (k, i) = s*A (k, j) + c*B (k, i);
      A (k, j) = tau;
    end;
  end;
  for j = 1: i - 1,
    rho = sqrt (B (i, i)^2 + B (i, j)^2);
    s = B (i, j)/rho;
    c = B (i, i)/rho;
    for k = 1: i,
      tau = c*B (k, j) - s*B (k, i);
      B (k, i) = s*B (k, j) + c*B (k, i);
      B (k, j) = tau;
    end;
  end;
end;
```

Computational complexity:

$\frac{2}{3}n^2(2n + 3r + 6) + 6nr + \frac{8}{3}n$ flops $+ \frac{1}{2}n(n + 2r + 1)\sqrt{}$.

$n \times (n + r)$ matrix. For the method of Dyer and McReynolds [78], this matrix is composed of two blocks containing the matrices $GC_Q(n \times r)$ and $\Phi P(n \times n)$. The specialization of the Householder algorithm to use the matrices GC_Q and ΦP directly, without having to place them into a common array first, is described and listed in Table 6.14. Algorithms for computing GC_Q and ΦP in place were given in the previous subsection.

6.5.2 Bierman–Thornton *UD* Filtering

This is a pair of algorithms, including the Bierman algorithm for the observational update of the modified Cholesky factors U and D of the covariance matrix $P = UDU^{\mathrm{T}}$ and the corresponding Thornton algorithm for the temporal update of U and D.

6.5.2.1 Bierman* UD *Observational Update Bierman's algorithm is one of the more stable implementations of the Kalman filter observational update. It is similar

TABLE 6.14 Schmidt–Householder Temporal Update Algorithm

This modification of the Householder algorithm performs upper triangularization of the partitioned matrix $[\Phi C_P(+), GC_Q]$ by modifying $\Phi C_P(+)$ and GC_Q in place using Householder transformations of the (effectively) partitioned matrix.

Input Variables

Symbol	Description
A	$n \times n$ matrix $\Phi\ C_P(+)$
B	$n \times r$ matrix GC_Q

Output Variables

A	Array is overwritten by upper triangular result $C_P(-)$ such that $C_P(-)C_P^{\mathrm{T}}(-) = \Phi C_P(+)C_P^{\mathrm{T}}(+)\Phi^{\mathrm{T}} + GC_QC_Q^{\mathrm{T}}G^{\mathrm{T}}$.
B	Array is zeroed during processing

Intermediate Variables

α, β, σ	Scalars
v	Scratchpad n-vector
w	Scratchpad $(n + r)$-vector

```
for k = n: -1:1,
  sigma = 0;
  for j = 1: r,
    sigma = sigma + B (k,j)^2;
  end;
  for j = 1: k,
    sigma = sigma + A (k,j)^2;
  end;
  alpha = sqrt (sigma);
  sigma = 0;
  for j = 1: r,
    w (j) = B (k,j);
    sigma = sigma + w (j)^2;
  end;
  for j = 1: k,
    if j == k
      v (j) = A (k,j) - alpha;
    else
      v (j) = A (k,j);
    end;
    sigma = sigma + v (j)^2;
  end;
  alpha = 2/sigma;
  for i = 1: k,
    sigma = 0;
    for j = 1: r,
```

(*Continued*)

TABLE 6.14 *Continued*

```
      sigma = sigma + B (i,j)*w (j);
    end;
    for j = 1: k,
      sigma = sigma + A (i,j)*v (j);
    end;
    beta = alpha*sigma;
    for j = 1: r,
      B (i,j) = B (i,j) - beta*w (j);
    end;
    for j = 1: k,
      A (i,j) = A (i,j) - beta*v(j);
    end;
  end;
end;
```

Computational complexity: $n^3r + \frac{1}{2}(n+1)^2r + 5 + \frac{1}{3}(2n+1)$ flops.

in form and computational complexity to the Carlson algorithm but avoids taking scalar square roots. (It has been called *square-root filtering without square roots*.) The algorithm was developed by the late Gerald J. Bierman (1941–1987), who made many useful contributions to optimal estimation theory, especially in implementation methods.

Partial UD *Factorization of Covariance Equations* In a manner similar to the case with Cholesky factors for scalar-valued measurements, the conventional form of the observational update of the covariance matrix

$$P(+) = P(-) - \frac{P(-)H^{\mathrm{T}}HP(-)}{R + HP(-)H^{\mathrm{T}}}$$

can be partially factored in terms of UD factors:

$$P(-) \stackrel{\text{def}}{=} U(-)D(-)U^{\mathrm{T}}(-), \tag{6.74}$$

$$P(+) \stackrel{\text{def}}{=} U(+)D(+)U^{\mathrm{T}}(+), \tag{6.75}$$

$$\begin{aligned} U(+)D(+)U^{\mathrm{T}}(+) &= U(-)D(-)U^{\mathrm{T}}(-) \\ &\quad - \frac{U(-)D(-)U^{\mathrm{T}}(-)H^{\mathrm{T}}HU(-)D(-)}{R + HU(-)D(-)U^{\mathrm{T}}(-)H^{\mathrm{T}}}U^{\mathrm{T}}(-) \end{aligned} \tag{6.76}$$

$$= U(-)D(-)U^{\mathrm{T}}(-) - \frac{U(-)D(-)\mathbf{v}\mathbf{v}^{\mathrm{T}}D(-)U^{\mathrm{T}}(-)}{R + \mathbf{v}^{\mathrm{T}}D(-)\mathbf{v}} \tag{6.77}$$

$$= U(-)\left[D(-) - \frac{D(-)\mathbf{v}\mathbf{v}^{\mathrm{T}}D(-)}{R + \mathbf{v}^{\mathrm{T}}D(-)\mathbf{v}}\right]U^{\mathrm{T}}(-), \tag{6.78}$$

where

$$\mathbf{v} = U^{\mathrm{T}}(-)H^{\mathrm{T}} \tag{6.79}$$

is an n-vector and n is the dimension of the state vector.

Equation 6.78 contains the unfactored expression

$$D(-) - D(-)\mathbf{v}[\mathbf{v}^{\mathrm{T}}D(-)\mathbf{v} + R]^{-1}\mathbf{v}^{\mathrm{T}}D(-).$$

If one were able to factor it with a unit triangular factor B in the form

$$D(-) - D(-)\mathbf{v}[\mathbf{v}^{\mathrm{T}}D(-)\mathbf{v} + R]^{-1}\mathbf{v}^{\mathrm{T}}D(-) = BD(+)B^{\mathrm{T}}, \tag{6.80}$$

then $D(+)$ would be the *a posteriori* D factor of P because the resulting equation

$$U(+)D(+)U^{\mathrm{T}}(+) = U(-)\{BD(+)B^{\mathrm{T}}\}U^{\mathrm{T}}(-) \tag{6.81}$$

$$= \{U(-)B\}D(+)\{U(-)B\}^{\mathrm{T}} \tag{6.82}$$

can be solved for the *a posteriori* U factor as

$$U(+) = U(-)B. \tag{6.83}$$

Therefore, for the observational update of the *UD* factors of the covariance matrix P, it suffices to find a numerically stable and efficient method for the *UD* factorization of a matrix expression of the form $D - D\mathbf{v}[\mathbf{v}^{\mathrm{T}}D\mathbf{v} + R]^{-1}\mathbf{v}^{\mathrm{T}}D$, where $\mathbf{v} = U^{\mathrm{T}}H^{\mathrm{T}}$ is a column vector. Bierman [31] found such a solution in terms of a rank 1 modification algorithm for modified Cholesky factors.

Bierman UD *Factorization* Derivations of the Bierman *UD* observational update can be found in [31]. It is implemented in the accompanying MATLAB m-file `bierman.m`.

An alternative algorithm implementing the Bierman *UD* observational update in place is given in Table 6.15. One can also store $\mathbf{w}$ over $\mathbf{v}$ to save memory requirements. It is possible to reduce the memory requirements even further by storing D on the diagonal of U or, better yet, storing just the strictly upper triangular part of U in a singly subscripted array. These programming tricks do little to enhance readability of the algorithms, however. They are best avoided unless one is truly desperate for memory.

6.5.2.2 Thornton* UD *Temporal Update This *UD* factorization of the temporal update in the discrete-time Riccati equation is due to Thornton [268]. It is also called *modified weighted Gram–Schmidt* (*MWGS*) orthogonalization.[7] It uses

[7]Gram–Schmidt *orthonormalization* is a procedure for generating a set of "unit normal" vectors as linear combinations of a set of linearly independent vectors. That is, the resulting vectors are mutually orthogonal and have unit length. The procedure without the unit-length property is called Gram–Schmidt

TABLE 6.15 Bierman Observational Update

Symbol	Definition
z	Value of scalar measurement
R	Variance of scalar measurement uncertainty
H	Scalar measurement sensitivity vector ($1 \times n$ matrix)
U, D	UD factors of $P(-)$ (input) and $P(+)$ (output)
x	State estimates $x(-)$ (input) and $x(+)$ (output)
v	scratchpad n-vector
w	scratchpad n-vector

```
delta = z;
for j = 1: n,
  delta = delta - H (j)*x (j);
  v (j) = H (j);
  for i = 1: j - 1,
    v (j) = v (j) + U (i, j)*H (i);
  end;
end;
sigma = R;
for j = 1:n,
  nu = v (j);
  v (j) = v (j)*D (j, j);
  w (j) = nu;
  for i = 1: j - 1,
    tau = U (i, j)*nu;
    U (i, j) = U (i, j) - nu*w (i)/sigma;
    w (i) = w (i) + tau;
  end;
  D (j, j) = D (j, j)*sigma;
  sigma = sigma + nu*v (j);
  D (j, j) = D (j, j)*sigma;
end;
epsilon = delta/sigma;
for i = 1: n,
  x (i) = x (i) + v (i)*epsilon;
end;
```

Computational complexity: $(2n^2 + 7n + 1)$ flops.

a factorization algorithm due to Björck [38] that is actually quite different from the conventional Gram–Schmidt orthogonalization algorithm and is more robust against roundoff errors. However, the algebraic properties of Gram–Schmidt orthogonalization are useful for deriving the factorization.

orthogonalization. These algorithmic methods were derived by Jorgen Pedersen Gram (1850–1916) and Erhard Schmidt (1876–1959).

Gram–Schmidt orthogonalization is an algorithm for finding a set of n mutually orthogonal m-vectors $b_1, b_2, \mathrm{b}_3, \ldots, b_m$ that are linear combinations of a set of n *linearly independent* m-vectors $a_1, a_2, a_3, \ldots, a_\mathrm{m}$. That is, the inner products

$$b_i^\mathrm{T} b_j = \begin{cases} |b_i|^2 & \text{if } i = j, \\ 0 & \text{if} i \neq j. \end{cases} \tag{6.84}$$

The Gram–Schmidt algorithm defines a unit lower[8] triangular $n \times n$ matrix L such that $A = BL$, where

$$A = \begin{bmatrix} a_1 & a_2 & a_3 & \ldots & a_n \end{bmatrix} \tag{6.85}$$

$$= BL \tag{6.86}$$

$$= \begin{bmatrix} b_1 & b_2 & b_3 & \ldots & b_n \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & \ldots & 0 \\ \ell_{21} & 1 & 0 & \ldots & 0 \\ \ell_{31} & l_{32} & 1 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \ell_{n1} & \ell_{n2} & \ell_{n3} & \ldots & 1 \end{bmatrix}, \tag{6.87}$$

where the vectors b_i are column vectors of B and the matrix product

$$B^\mathrm{T} B = \begin{bmatrix} |b_1|^2 & 0 & 0 & \ldots & 0 \\ 0 & |b_2|^2 & 0 & \ldots & 0 \\ 0 & 0 & |b_3|^2 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & |b_n|^2 \end{bmatrix} \tag{6.88}$$

$$= \mathrm{diag}_{1 \leq i \leq n} \{|b_i|^2\} \tag{6.89}$$

$$= D_\beta, \tag{6.90}$$

a diagonal matrix with positive diagonal values $\beta_i = |b_i|^2$.

Weighted Gram–Schmidt Orthogonalization The m-vectors x and y are said to be orthogonal *with respect to the weights* $w_1, w_2, w_3, \ldots, w_m$ if the

[8]In its original form, the algorithm produced a unit upper triangular matrix U by processing the a_i in the order $i = 1, 2, 3, \ldots, n$. However, if the order is reversed, the resulting coefficient matrix will be lower triangular and the resulting vectors b_i will still be mutually orthogonal.

weighted inner product

$$\sum_{i=1}^{m} x_i w_i y_i = x^{\mathrm{T}} D_w y \tag{6.91}$$

$$= 0, \tag{6.92}$$

where the diagonal *weighting matrix*

$$D_w = \mathrm{diag}_{1 \le i \le n}\{w_i\}. \tag{6.93}$$

The Gram–Schmidt orthogonalization procedure can be extended to include orthogonality of the column vectors b_i and b_j with respect to the weighting matrix D_w:

$$b_i^{\mathrm{T}} D_w b_j = \begin{cases} \beta_i > 0 & \text{if } i = j, \\ 0 & \text{if } i \neq j. \end{cases} \tag{6.94}$$

The resulting weighted Gram–Schmidt orthogonalization of a set of n linearly independent m-vectors $a_1, a_2, a_3, \ldots, a_m$ with respect to a weighting matrix D_w defines a unit lower triangular $n \times n$ matrix L_w such that the product $AL_w = B_w$ and

$$B_w^{\mathrm{T}} D_w B_w = \mathrm{diag}_{1 \le i \le n}\{\beta_i\}, \tag{6.95}$$

where $D_w = I$ for conventional orthogonalization.

Modified Weighted Gram–Schmidt Orthogonalization The standard Gram–Schmidt orthogonalization algorithms are not reliably stable numerical processes when the column vectors of A are close to being linearly dependent or the weighting matrix has a large condition number. An alternative algorithm due to Björck [38] has better overall numerical stability. Although L is not an important result for the orthogonalization problem (B is), its inverse turns out to be more useful for the *UD* filtering problem.

The *Thornton temporal update for UD factors* uses triangularization of the Q matrix (if it is not already diagonal) in the form $Q = GD_Q G^{\mathrm{T}}$, where D_Q is a diagonal matrix. If we let the matrices

$$A = \begin{bmatrix} U_{k-1}^{\mathrm{T}}(+)\Phi_k^{\mathrm{T}} \\ G_k^{\mathrm{T}} \end{bmatrix}, \tag{6.96}$$

$$D_w = \begin{bmatrix} D_{k-1}(+) & 0 \\ 0 & D_{Q_k} \end{bmatrix}, \tag{6.97}$$

then the MWGS orthogonalization procedure will produce a unit lower triangular $n \times n$ matrix L^{-1} and a diagonal matrix D_β such that

$$A = BL, \tag{6.98}$$

$$L^{\mathrm{T}} D_\beta L = L^{\mathrm{T}} B^{\mathrm{T}} D_w B L \tag{6.99}$$

$$= (BL)^{\mathrm{T}} D_w B L \tag{6.100}$$

$$= A^{\mathrm{T}} D_w A \tag{6.101}$$

$$= \begin{bmatrix} \Phi_{k-1} U_{k-1}(+) & G_{k-1} \end{bmatrix} \begin{bmatrix} D_{k-1}(+) & 0 \\ 0 & D_{Q_{k-1}} \end{bmatrix} \begin{bmatrix} U_{k-1}^{\mathrm{T}}(+) \Phi_{k-1}^{\mathrm{T}} \\ G_{k-1}^{\mathrm{T}} \end{bmatrix} \tag{6.102}$$

$$= \Phi_{k-1} U_{k-1}(+) D_{k-1}(+) U_{k-1}^{\mathrm{T}}(+) \Phi_{k-1}^{\mathrm{T}} + G_{k-1} D_{Q_{k-1}} G_{k-1}^{\mathrm{T}} \tag{6.103}$$

$$= \Phi_{k-1} P_{k-1}(+) \Phi_{k-1}^{\mathrm{T}} + Q_{k-1} \tag{6.104}$$

$$= P_k(-). \tag{6.105}$$

Consequently, the factors

$$U_k(-) = L^{\mathrm{T}}, \tag{6.106}$$

$$D_k(-) = D_\beta \tag{6.107}$$

are the solutions of the temporal update problem for the *UD* filter.

Diagonalizing Q It is generally worth the effort to "diagonalize" Q (if it is not already a diagonal matrix), because the net computational complexity is reduced by this procedure. The formula given for total computational complexity of the Thornton algorithm in Table 6.16 includes the computational complexity for the *UD* decomposition of Q as $U_Q \acute{Q} U_Q^{\mathrm{T}}$ for $\acute{Q}$ diagonal $\left[\frac{1}{6} p(p-1)(p+4) \text{ flops}\right]$ plus the computational complexity for multiplying G by the resulting $p \times p$ unit upper triangular factor U_Q $[\frac{1}{2} np(p-1) \text{ flops}]$ to obtain $\acute{G}$.

The algorithm listed in Table 6.16 operates with the matrix blocks ΦU, $\acute{G}$, D, and $\acute{Q}$ by name and not as submatrices of some larger matrix. It is not necessary to physically relocate these matrices into larger arrays in order to apply the Björck orthogonalization procedure. The algorithm is written to find them in their original arrays. (It makes the listing a little longer, but the computational complexity is the same.)

Multiplying Φ U *in Place* The complexity formulas in Table 6.16 also include the computations required for forming the product ΦU of the $n \times n$ state transition matrix

TABLE 6.16 Thornton *UD* Temporal Update Algorithm*

Symbol	Description
	Inputs
D	The $n \times n$ diagonal matrix. Can be stored as an n-vector.
ΦU	Matrix product of $n \times n$ state transition matrix Φ and $n \times n$ unit upper triangular matrix U such that $UDU^{\mathrm{T}} = P(+)$, the covariance matrix of *a posteriori* state estimation uncertainty.
$\acute{G}$	$= GU_Q$. The modified $n \times p$ process noise coupling matrix, where $Q = U_Q D_Q U_Q^{\mathrm{T}}$
D_Q	Diagonalized $p \times p$ covariance matrix of process noise. Can be stored as a p-vector.
	Outputs
$\Phi\ U$	Is overwritten by intermediate results.
$\acute{G}$	Is overwritten by intermediate results.
$\acute{D}$	The $n \times n$ diagonal matrix. Can be stored as an n-vector.
$\acute{U}$	The $n \times n$ unit upper triangular matrix such that $\acute{U}\acute{D}\acute{U}^{\mathrm{T}} = \Phi UDU^{\mathrm{T}}\Phi^{\mathrm{T}} + GQG^{\mathrm{T}}$.

```
for i = n: -1:1,
  sigma = 0;
  for j = 1: n,
    sigma = sigma + PhiU (i, j)^ 2*D (j, j);
  end;
  for j = 1: p,
    sigma = sigma + G (i, j)^ 2*DQ (j, j);
  end;
  D (i, i) = sigma;
  U (i, i) = 1;
  for j = 1: i - 1,
    sigma = 0;
    for k = 1: n,
      sigma = sigma + PhiU (i, k)*D (k, k)*PhiU (j, k);
    end;
    for k = 1: p,
      sigma = sigma + G (i, k)*DQ (k, k)*G (j, k);
    end;
    U (j, i) = sigma/D (i, i);
    for k = 1: n,
      PhiU (j, k) = PhiU (j, k) - U (j, i)*PhiU (i, k);
    end;
    for k = 1: p,
      G (j, k) = G (j, k) - U (j, i)*G (i, k);
    end;
  end;
end;
```

(*Continued*)

TABLE 6.16 *Continued*

Computational Complexity Breakdown (in flops)	
Operation	flops
Matrix product ΦU	$n^2(n-1)/2$
Solve $U_Q D_Q U_Q^{\mathrm{T}} = Q, \acute{G} = GU_Q$	$p(p-1)(3n+p+4)/6$
Thornton algorithm	$3n(n-1)(n+p)/2$
Total	$n(n-1)(4n+3p-1)/2 + p(p-1)(3n+p+4)/6$

*Performs the temporal update of the modified Cholesky factors (*UD* factors) of the covariance matrix of state estimation uncertainty for the Kalman filter.

Φ and the $n \times n$ unit upper triangular matrix U. This matrix product can be computed in place—overwriting Φ with the product Φ U—by the following algorithm:

```
for i = 1 : n,
     for j = n : -1:1,
        sigma = Phi (i, j);
        for k = 1 : j - i,
           sigma = sigma + Phi (i, k)*U (k, j);
        end;
     Phi (i, j) = sigma;
     end;
end;
```

The computational complexity of this specialized matrix multiplication algorithm is $n^2(n-1)/2$, which is less than half of the computational complexity of the general $n \times n$ matrix product (n^3). Performing this multiplication in place also frees up the array containing $U_k(+)$ to accept the updated value $U_{k+1}(-)$. In some applications, Φ will have a sparse structure that can be exploited to reduce the computational requirements even more.

6.6 OTHER IMPLEMENTATION METHODS

The main thrust of this chapter is the square-root filtering methods presented in the previous section. Although factorization methods are probably the most numerically stable implementation methods currently available, there are alternative methods that may perform adequately on some applications, and there are some earlier alternatives that bear mentioning for their historical significance and for comparisons with the factorization methods.

6.6.1 Earlier Implementation Methods

6.6.1.1 Swerling Inverse Formulation This is *not* recommended as an implementation method for the Kalman filter, because its computational complexity and numerical stability place it at a disadvantage relative to the other methods. Its computational complexity is derived here for the purpose of demonstrating this.

TABLE 6.17 Operation Summary for Swerling Inverse Formulation

Operation	Flops
R^{-1}	$\ell^3 + \frac{1}{2}\ell^2 + \frac{1}{2}\ell$ if $\ell > 1$, 1 if $\ell = 1$
$(R^{-1})H$	$n\ell^2$
$H^{\mathrm{T}}(R^{-1}H)$	$\frac{1}{2}n^2\ell + \frac{1}{2}n\ell$
$P^{-1}(-) + (H^{\mathrm{T}}R^{-1}H)$	$n^3 + \frac{1}{2}n^2 + \frac{1}{2}n$
$[P^{-1}(-) + H^{\mathrm{T}}R^{-1}H]^{-1}$	$n^3 + \frac{1}{2}n^2 + \frac{1}{2}n$
Total	$2n^3 + n^2 + n + \frac{1}{2}n^2\ell + n\ell^2 + \frac{1}{2}n\ell + \ell^3 + \frac{1}{2}\ell^2 + \ell$

Recursive Least Mean Squares This form of the covariance update for recursive least mean squares estimation was published by Swerling [265]. Swerling's estimator was essentially the Kalman filter but with the observational update equation for the covariance matrix in the form

$$P(+) = [P^{-1}(-) + H^{\mathrm{T}}R^{-1}H]^{-1}.$$

This implementation requires three matrix inversions and two matrix products. If the observation is scalar valued ($m = 1$), the matrix inversion R^{-1} requires only one divide operation. One can also take advantage of diagonal and symmetric matrix forms to make the implementation more efficient.

Computational Complexity of Swerling Inverse Formulation[9] For the case where the state dimension $n = 1$ and the measurement dimension $\ell = 1$, it can be done with four flops. In the case where $n > 1$, the dependence of the computational complexity on ℓ and n is shown in Table 6.17.[10] This is the most computationally intensive method of all. The number of arithmetic operations increases as n^3. The numbers of operations for the other methods increase as $n^2\ell + \ell^3$, but usually $n > \ell$ in Kalman filtering applications.

6.6.1.2 Kalman Formulation

Data Flows A data flow diagram of the implementation equations defined by Kalman [143] is shown in Figure 6.8. This shows the points at which the measurements (z) and model parameters (H, R, Φ, and Q) are introduced in the matrix operations. There is some freedom to choose exactly how these operations shown will be implemented, however, and the computational requirements can be reduced substantially by reuse of repeated subexpressions.

[9]See Section 6.4.3.6 for an explanation of how computational complexity is determined.

[10]There is an alternative matrix inversion method (due to Strassen [260]) that reduces the number of multiplies in an $n \times n$ matrix inversion from n^3 to $n \log_2 7$ but increases the number of additions significantly.

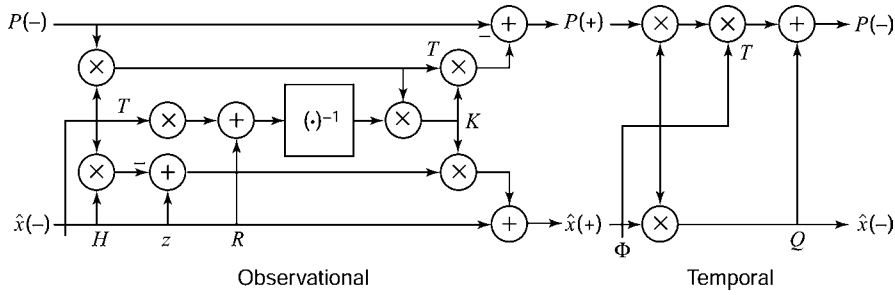


Figure 6.8 Data flows of Kalman update implementations.

Reuse of Partial Results In this conventional form of the observational update equations, the factor $\{HP\}$ occurs several times in the computation of $\overline{K}$ and P:

$$\overline{K} = P(-)H^{\mathrm{T}}[HP(-)H^{\mathrm{T}} + R]^{-1} \tag{6.108}$$

$$= \{HP(-)\}^{\mathrm{T}}[\{HP(-)\}H^{\mathrm{T}} + R]^{-1}, \tag{6.109}$$

$$P(+) = P(-) - \overline{K}\{HP(-)\}. \tag{6.110}$$

The factored form shows the reusable partial results $[HP(-)]$ and $\overline{K}$ (the Kalman gain). Using these partial results where indicated, the implementation of the factored form requires four matrix multiplies and one matrix inversion. As in the case of the Swerling formulation, the matrix inversion can be accomplished by a divide if the dimension of the observation (ℓ) is 1, and the efficiency of all operations can be improved by computing only the unique elements of symmetric matrices. The floating-point arithmetic operations required for the observational update of the Kalman filter, using these methods, are summarized in Table 6.18. Note that the total number of operations required does not increase as n^3, as is the case with the Swerling formulation.

6.6.1.3 Potter Square-Root Filter The inventor of square-root filtering is James E. Potter. Soon after the introduction of the Kalman filter, Potter introduced the idea of factoring the covariance matrix and provided the first of the square-root

TABLE 6.18 Operation Summary for the Conventional Kalman Filter

Operation	Flops
$H \times P(-)$	$n^2\ell$
$H \times [HP(-)]^{\mathrm{T}} + R$	$\frac{1}{2}n\ell^2 + \frac{1}{2}n\ell$
$\{H[HP(-)]^{\mathrm{T}} + R\}^{-1}$	$\ell^3 + \frac{1}{2}\ell^2 + \frac{1}{2}\ell$
$\{H[HP(-)]^{\mathrm{T}} + R\}^{-1} \times [HP(-)]$	$n\ell^2$
$P(-) - [HP(-)] \times \{H[HP(-)]^{\mathrm{T}} + R\}^{-1}[HP(-)]$	$\frac{1}{2}n^2\ell + \frac{1}{2}n\ell$
Total	$\frac{3}{2}n^2\ell + \frac{3}{2}n\ell^2 + n\ell + \ell^3 + \frac{1}{2}\ell^2 + \frac{1}{2}\ell$

methods for the observational update (see Battin [26, pp. 338–340] and Potter and Stern [217]).

Potter defined the Cholesky factor of the covariance matrix P as

$$P(-) \stackrel{\text{def}}{=} C(-)C^{\mathrm{T}}(-), \tag{6.111}$$

$$P(+) \stackrel{\text{def}}{=} C(+)C(+)^{\mathrm{T}} \tag{6.112}$$

so that the observational update equation

$$P(+) = P(-) - P(-)H^{\mathrm{T}}[HP(-)H^{\mathrm{T}} + R]^{-1}HP(-) \tag{6.113}$$

could be partially factored as

$$\begin{aligned} C(+)C^{\mathrm{T}}(+) &= C(-)C^{\mathrm{T}}(-) \\ &\quad - C(-)C^{\mathrm{T}}(-)H^{\mathrm{T}}[HC(-)C^{\mathrm{T}}(-)H^{\mathrm{T}} + R]^{-1}HC(-)C^{\mathrm{T}}(-) && (6.114) \\ &= C(-)C^{\mathrm{T}}(-) - C(-)V[V^{\mathrm{T}}V + R]^{-1}V^{\mathrm{T}}C^{\mathrm{T}}(-) && (6.115) \\ &= C(-)\{I_n - V[V^{\mathrm{T}}V + R]^{-1}V^{\mathrm{T}}\}C^{\mathrm{T}}(-), && (6.116) \end{aligned}$$

where

$I_n = n \times n$ identity matrix
$V = C^{\mathrm{T}}(-)H^{\mathrm{T}}$ is an $n \times \ell$ general matrix
$n =$ dimension of state vector
$\ell =$ dimension of measurement vector

Equation 6.116 contains the unfactored expression $\{I_n - V[V^{\mathrm{T}}V + R]^{-1}V^{\mathrm{T}}\}$. For the case where that measurement is a scalar ($\ell = 1$), Potter was able to factor it in the form

$$I_n - V[V^{\mathrm{T}}V + R]^{-1}V^{\mathrm{T}} = WW^{\mathrm{T}} \tag{6.117}$$

so that the resulting equation

$$\begin{aligned} C(+)C(+)^{\mathrm{T}} &= C(-)\{WW^{\mathrm{T}}\}C^{\mathrm{T}}(-) && (6.118) \\ &= \{C(-)W\}\{C(-)W\}^{\mathrm{T}} && (6.119) \end{aligned}$$

could be solved for the *a posteriori* Cholesky factor of $P(+)$ as

$$C(+) = C(-)W. \tag{6.120}$$

When the measurement is a scalar, the expression to be factored is a symmetric *elementary matrix* of the form[11]

$$I_n - \frac{\mathbf{v}\mathbf{v}^{\mathrm{T}}}{R + |\mathbf{v}|^2}, \tag{6.121}$$

where R is a positive scalar and $\mathbf{v} = C^{\mathrm{T}}(-)H^{\mathrm{T}}$ is a column n-vector.

The formula for the symmetric square root of a symmetric elementary matrix is given in Equation 6.44. For the elementary matrix format in Equation 6.112, the scalar s of Equation 6.44 has the value

$$s = \frac{1}{R + |\mathbf{v}|^2}, \tag{6.122}$$

so that the radicand

$$1 - s|\mathbf{v}|^2 = 1 - \frac{|\mathbf{v}|^2}{R + |\mathbf{v}|^2} \tag{6.123}$$

$$= \frac{R}{R + |\mathbf{v}|^2} \tag{6.124}$$

$$\geq 0 \tag{6.125}$$

because the variance $R \geq 0$. Consequently, the matrix expression 6.112 will always have a real matrix square root.

Potter Formula for Observational Updates Because the matrix square roots of symmetric elementary matrices are also symmetric matrices, they are also Cholesky factors. That is,

$$(I - s\,\mathbf{v}\mathbf{v}^{\mathrm{T}}) = (I - \sigma\mathbf{v}\mathbf{v}^{\mathrm{T}})(I - \sigma\mathbf{v}\mathbf{v}^{\mathrm{T}}) \tag{6.126}$$

$$= (I - \sigma\mathbf{v}\mathbf{v}^{\mathrm{T}})(I - \sigma\mathbf{v}\mathbf{v}^{\mathrm{T}})^{\mathrm{T}}. \tag{6.127}$$

[11]This expression—or something very close to it—is used in many of the square-root filtering methods for observational updates. The Potter square-root filtering algorithm finds a symmetric factor W, which does not preserve triangularity of the product $C(-)W$. The Carlson observational update algorithm (in Section 6.5.1.1) finds a triangular factor W, which preserves the triangularity of $C(+) = C(-)W$ if both factors $C(+)$ and W are of the same triangularity (i.e., if both $C(-)$ and W are upper triangular or both are lower triangular). The Bierman observational update algorithm uses a related UD factorization. Because the rank of the matrix $\mathbf{v}\mathbf{v}^{\mathrm{T}}$ is 1, these methods are referred to as *rank 1 modification methods*.

Following the approach leading to Equation 6.120, the solution for the *a posteriori* Cholesky factor $C(+)$ of the covariance matrix P can be expressed as the product

$$C(+)C^{\mathrm{T}}(+) = P(+) \tag{6.128}$$

$$= C(-)(I - s\mathbf{v}\mathbf{v}^{\mathrm{T}})C^{\mathrm{T}}(-) \tag{6.129}$$

$$= C(-)(I - \sigma\mathbf{v}\mathbf{v}^{\mathrm{T}})(I - \sigma\mathbf{v}\mathbf{v}^{\mathrm{T}})^{\mathrm{T}}C^{\mathrm{T}}(-), \tag{6.130}$$

which can be factored as[12]

$$C(+) = C(-)(I - \sigma\mathbf{v}\mathbf{v}^{\mathrm{T}}) \tag{6.131}$$

with

$$\sigma = \frac{1 + \sqrt{1 - s|\mathbf{v}|^2}}{|\mathbf{v}|^2} \tag{6.132}$$

$$= \frac{1 + \sqrt{R/(R + |\mathbf{v}|^2)}}{|\mathbf{v}|^2}. \tag{6.133}$$

Equations 6.131 and 6.133 define the Potter square-root observational update formula, which is implemented in the accompanying MATLAB m-file `potter.m`. The Potter formula can be implemented *in place* (i.e., by overwriting C).

This algorithm updates the state estimate x and a Cholesky factor C of P in place. This Cholesky factor is a general $n \times n$ matrix. That is, it is not maintained in any particular form by the Potter update algorithm. The other square-root algorithms maintain C in triangular form.

6.6.1.4 Joseph-Stabilized Implementation This variant of the Kalman filter is due to Bucy and Joseph [56], who demonstrated improved numerical stability by rearranging the standard formulas for the observational update (given here for scalar measurements) into the formats

$$\acute{z} = R^{-1/2}z, \tag{6.134}$$

$$\acute{H} = \acute{z}H, \tag{6.135}$$

$$\overline{K} = (\acute{H}P(-)\acute{H}^{\mathrm{T}} + 1)^{-1}P(-)\acute{H}^{\mathrm{T}}, \tag{6.136}$$

$$P(+) = (I - \overline{K}\acute{H})P(-)(I - \overline{K}\acute{H})^{\mathrm{T}} + \overline{K}\overline{K}^{\mathrm{T}}, \tag{6.137}$$

taking advantage of partial results and the redundancy due to symmetry. The mathematical equivalence of Equation 6.137 to the conventional update formula for the covariance matrix was shown as Equation 4.23. This formula, by itself, does not

[12]Note that, as $R \to \infty$ (no measurement), $\sigma \to 2/|\mathbf{v}|^2$ and $I - \sigma\mathbf{v}\mathbf{v}^{\mathrm{T}}$ becomes a Householder matrix.

uniquely define the Joseph implementation, however. As shown, it has $\sim n^3$ computational complexity.

Bierman Implementation This is a slight alteration due to Bierman [31] that reduces computational complexity by measurement decorrelation (if necessary) and the parsimonious use of partial results. The data flow diagram shown in Figure 6.9 is for a scalar measurement update, with data flow from top (inputs) to bottom (outputs) and showing all intermediate results. Calculations at the same level in this diagram may be implemented in parallel. Intermediate (temporary) results are labeled as T_1, T_2, ..., T_8, where $T_6 = \bar{K}$, the Kalman gain. If the result (left-hand side) of an $m \times m$ array is symmetric, then only the $\frac{1}{2}m(m+1)$ unique elements need be computed. Bierman [31] has made the implementation more memory efficient by the reuse of memory locations for these intermediate results. Bierman's implementation does not eliminate the redundant memory from symmetric arrays, however.

The computational complexity of this implementation grows as $3\ell n(3n+5)/2$ flops, where n is the number of components in the state vector and ℓ is the number

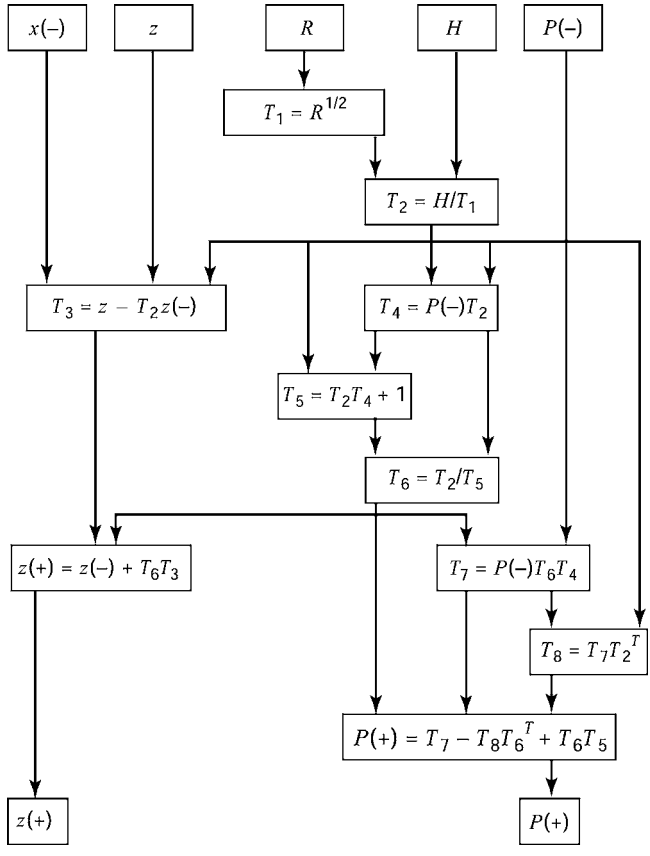


Figure 6.9 Data flow of the Bierman–Joseph implementation.

of components in the measurement vector [31]. However, this formulation does require that R be a diagonal matrix. Otherwise, additional computational complexity of $(4\ell^3 + \ell^2 - 10\ell + 3\ell^2 n - 3\ell n)/6$ flops for measurement decorrelation is incurred.

De Vries Implementation This implementation, which was shown to the authors by Thomas W. De Vries at Rockwell International, is designed to reduce the computational complexity of the Joseph formulation by judicious rearrangement of the matrix expressions and reuse of intermediate results. The fundamental operations—and their computational complexities—are summarized in Table 6.19.

Negative Evaluations of Joseph-Stabilized Implementation In comparative evaluations of several Kalman filter implementations on orbit estimation problems, Thornton and Bierman [269] found that the Joseph-stabilized implementation failed on some ill-conditioned problems for which square-root methods performed well.

6.6.2 Morf–Kailath Combined Observational/Temporal Update

The lion's share of the computational effort in Kalman filtering is spent in solving the Riccati equation. This effort is necessary for computing the Kalman gains. However, only the *a priori* value of the covariance matrix is needed for this purpose. Its *a*

TABLE 6.19 De Vries–Joseph Implementation of Covariance Update

Operation	Complexity
Without Using Decorrelation	
$\mathcal{F}_1 = P(-)H^{\mathrm{T}}$	ℓn^2
$\mathcal{F}_2 = H\mathcal{F}\mathrm{F}_1 + R$	$n\ell(\ell+1)/2$
$\mathcal{U}\mathcal{D}\mathcal{U}^{\mathrm{T}} = \mathcal{F}_2$	$\frac{1}{6}\ell(\ell+1)(\ell+2)$ (UD factorization)
$\mathcal{U}\mathcal{D}\mathcal{U}^{\mathrm{T}}K^{\mathrm{T}} = \mathcal{F}_1^{\mathrm{T}}$	$\ell^2 n$ [to solve for K]
$\mathcal{F}_3 = \frac{1}{2}K\mathcal{F}_2 - \mathcal{F}_1$	$\ell^2(n+1)$
$\mathcal{F}_4 = \mathcal{F}_3 K^{\mathrm{T}}$	ℓn^2
$P(+) = P(-) + \mathcal{F}_4 + \mathcal{F}_4^{\mathrm{T}}$	(included above)
Total	$\frac{1}{6}\ell^3 + \frac{3}{2}\ell^2 + \frac{1}{3}\ell + \frac{1}{2}\ell n + \frac{5}{2}\ell^2 n + 2\ell n^2$
Using Decorrelation	
Decorrelation ℓ repeats:	$\frac{2}{3}\ell^3 + \ell^2 - \frac{5}{3}\ell - \frac{1}{2}\ell n + \frac{1}{2}\ell^2 n$
$\mathcal{F}_1 = P(-)H^{\mathrm{T}}$	n^2
$\mathcal{F}_2 = H\mathcal{F}_1 + R$	n
$K = \mathcal{F}_1/\mathcal{F}_2$	n
$\mathcal{F}_3 = \frac{1}{2}K\mathcal{F}_2 - \mathcal{F}_1$	$n+1$
$\mathcal{F}_4 = \mathcal{F}_3 K^{\mathrm{T}}$	n^2
$P(+) = P(-) + \mathcal{F}_4 + \mathcal{F}_4^{\mathrm{T}}$	(included above)
Total	$\frac{2}{3}\ell^3 + \ell^2 - \frac{2}{3}\ell + \frac{5}{3}\ell n + \frac{1}{2}\ell^2 n + 2\ell n^2$

posteriori value is only used as an intermediate result on the way to computing the next *a priori* value.

Actually, it is not necessary to compute the *a posteriori* values of the covariance matrix explicitly. It is possible to compute the *a priori* values from one temporal epoch to the next temporal epoch, without going through the intermediate *a posteriori* values. This concept, and the methods for doing it, were introduced by Morf and Kailath [201].

6.6.2.1 Combined Updates of Cholesky Factors The direct computation of $C_{P(k+1)}(-)$, the triangular Cholesky factor of $P_{k+1}(-)$, from $C_{P(k)}(-)$, the triangular Cholesky factor of $P_k(-)$, can be implemented by triangularization of the $(n+m)\times(p+n+m)$ partitioned matrix

$$A_k = \begin{bmatrix} GC_{Q(k)} & \Phi_k C_{P(k)} & 0 \\ 0 & H_k C_{P(k)} & C_{R(k)} \end{bmatrix}, \tag{6.138}$$

where $C_{R(k)}$ is a Cholesky factor of R_k and $C_{Q(k)}$ is a Cholesky factor of Q_k. Note that the $(n+m)\times(n+m)$ symmetric product

$$A_k A_k^{\mathrm{T}} = \begin{bmatrix} \Phi_k P_k(-)\Phi_k^{\mathrm{T}} + G_k Q_k G_k^{\mathrm{T}} & \Phi_k P_k(-) H_k^{\mathrm{T}} \\ H_k P_k(-)\Phi_k^{\mathrm{T}} & H_k P_k(-) H_k^{\mathrm{T}} + R_k \end{bmatrix}. \tag{6.139}$$

Consequently, if A_k is upper triangularized in the form

$$A_k T = C_k \tag{6.140}$$

$$= \begin{bmatrix} 0 & C_{P(k+1)} & \Psi_k \\ 0 & 0 & C_{E(k)} \end{bmatrix} \tag{6.141}$$

by an orthogonal transformation T, then the matrix equation

$$C_k C_k^{\mathrm{T}} = A_k A_k^{\mathrm{T}}$$

implies that the newly created block submatrices $C_{E(k)}$,Ψ_k, and $C_{P(k+1)}$ satisfy the equations

$$C_{E(k)} C_{E(k)}^{\mathrm{T}} = H_k P_k(-) H_k^{\mathrm{T}} + R_k \tag{6.142}$$

$$= E_k, \tag{6.143}$$

$$\Psi_k \Psi_k^{\mathrm{T}} = \Phi_k P_k(-) H_k^{\mathrm{T}} E_k^{-1} H_k P_k(-)\Phi_k, \tag{6.144}$$

$$\Psi_k = \Phi_k P_k(-) H_k^{\mathrm{T}} C_{E_k}^{-T}, \tag{6.145}$$

$$C_{P(k+1)}C_{P(k+1)}^{\mathrm{T}} = \Phi_k P_k(-)\Phi_k^{\mathrm{T}} + G_k Q_k G_k^{\mathrm{T}} - \Psi_k \Psi_k^{\mathrm{T}} \tag{6.146}$$

$$= P_{k+1}(-), \tag{6.147}$$

and the Kalman gain can be computed as

$$\overline{K}_k = \Psi_k C_{E(k)}^{-1}. \tag{6.148}$$

The computation of C_k from A_k can be done by Householder or Givens triangularization.

6.6.2.2 Combined Updates of* UD *Factors This implementation uses the *UD* factors of the covariance matrices *P, R*, and *Q*,

$$P_k = U_{P(k)} D_{P(k)} U_{P(k)}^{\mathrm{T}}, \tag{6.149}$$

$$R_k = U_{R(k)} D_{R(k)} U_{R(k)}^{\mathrm{T}}, \tag{6.150}$$

$$Q_k = U_{Q(k)} D_{Q(k)} U_{Q(k)}^{\mathrm{T}}, \tag{6.151}$$

in the partitioned matrices

$$B_k = \begin{bmatrix} GU_{Q(k)} & \Phi_k U_{P(k)} & 0 \\ 0 & H_k U_{P(k)} & U_{R(k)} \end{bmatrix}, \tag{6.152}$$

$$D_k = \begin{bmatrix} D_{Q(k)} & 0 & 0 \\ 0 & D_{P(k)} & 0 \\ 0 & 0 & D_{R(k)} \end{bmatrix}, \tag{6.153}$$

which satisfy the equation

$$B_k D_k B_k^{\mathrm{T}} = \begin{bmatrix} \Phi_k P_k(-)\Phi_k^{\mathrm{T}} + G_k Q_k G_k^{\mathrm{T}} & \Phi_k P_k(-) H_k^{\mathrm{T}} \\ H_k P_k(-)\Phi_k^{\mathrm{T}} & H_k P_k(-) H_k^{\mathrm{T}} + R_k \end{bmatrix}. \tag{6.154}$$

The MWGS orthogonalization of the rows of B_k with respect to the weighting matrix D_k will yield the matrices

$$\acute{B}_k = \begin{bmatrix} U_{P(k+1)} & U_{\Psi(k)} \\ 0 & U_{E(k)} \end{bmatrix}, \tag{6.155}$$

$$\acute{D}_k = \begin{bmatrix} D_{P(k+1)} & 0 \\ 0 & D_{E(k)} \end{bmatrix}, \tag{6.156}$$

where $U_{P(k+1)}$ and $D_{P(k+1)}$ are the UD factors of $P_{k+1}(-)$, and

$$U_{\Psi(k)} = \Phi_k P_k(-) H_k^{\mathrm{T}} U_{E(k)}^{-\mathrm{T}} D_{E(k)}^{-1} \tag{6.157}$$

$$= \Phi_k \overline{K}_k U_{E(k)}. \tag{6.158}$$

Consequently, the Kalman gain

$$\overline{K}_k = \Phi_k^{-1} U_{\Psi(k)} U_{E(k)}^{-1} \tag{6.159}$$

can be computed as a by-product of the MWGS procedure as well.

6.6.3 Information Filtering

6.6.3.1 Information Matrix of an Estimate The inverse of the covariance matrix of estimation uncertainty is called the *information matrix*[13]:

$$Y \stackrel{\text{def}}{=} P^{-1}. \tag{6.160}$$

Implementations using Y (or its Cholesky factors) rather than P (or its Cholesky factors) are called *information filters*. (Implementations using P are also called *covariance filters*.)

6.6.3.2 Uses of Information Filtering

Problems without Prior Information Using the information matrix, one can express the idea that an estimation process may start with no *a priori* information whatsoever, expressed by

$$Y_0 = 0, \tag{6.161}$$

a matrix of zeros. An information filter starting from this condition will have absolutely no bias toward the *a priori* estimate. Covariance filters cannot do this.

One can also represent *a priori* estimates with no information in specified subspaces of state space by using information matrices with characteristic values equal to zero. In that case, the information matrix will have an eigenvalue—eigenvector decomposition of the form

$$Y_0 = \sum_i \lambda_i e_i e_i^{\mathrm{T}}, \tag{6.162}$$

[13]This is also called the *Fisher information matrix*, named after the English statistician Ronald Aylmer Fisher (1890–1962). More generally, for distributions with differentiable probability density functions, the information matrix is defined as the matrix of second-order derivatives of the logarithm of the probability density with respect to the variates. For Gaussian distributions, this equals the inverse of the covariance matrix.

where some of the eigenvalues $\lambda_i = 0$ and the corresponding eigenvectors e_i represent directions in state space with zero *a priori* information. Subsequent estimates will have no bias toward these components of the *a priori* estimate.

Information filtering cannot be used if P is singular, just as covariance filtering cannot be used if Y is singular. However, one may switch representations if both conditions do not occur simultaneously. For example, an estimation problem with zero initial information can be started with an information filter and then switched to a covariance implementation when Y becomes nonsingular. Conversely, a filtering problem with zero initial uncertainty may be started with a covariance filter, then switched to an information filter when P becomes nonsingular.

Robust Observational Updates The observational update of the *uncertainty matrix* is less robust against roundoff errors than the temporal update. It is more likely to cause the uncertainty matrix to become indefinite, which tends to destabilize the estimation feedback loop.

The observational update of the information matrix is more robust against roundoff errors. This condition is the result of a certain duality between information filtering and covariance filtering by which the algorithmic structures of the temporal and observational updates are switched between the two approaches. The downside of this duality is that the temporal update of the information matrix is less robust than the observational update against roundoff errors and is a more likely cause of degradation. Therefore, information filtering may not be a panacea for all conditioning problems, but in those cases for which the observational update of the uncertainty matrix is the culprit, information filtering offers a possible solution to the roundoff problem.

Disadvantages of Information Filtering The greatest objection to information filtering is the loss of "transparency" of the representation. Although information is a more practical concept than uncertainty for some problems, it can be more difficult to interpret its physical significance and to use it in our thinking. With a little practice, it is relatively easy to visualize how σ (the square root of variance) is related to probabilities and to express uncertainties as "3σ" values. One must invert the information matrix before one can interpret its values in this way.

Perhaps the greatest impediment to widespread acceptance of information filtering is the loss of physical significance of the associated state vector components. These are linear combinations of the original state vector components, but the coefficients of these linear combinations change with the state of information/uncertainty in the estimates.

6.6.3.3 Information States Information filters do not use the same state vector representations as covariance filters. Those that use the information matrix in the filter implementation use the *information state*

$$d \stackrel{\text{def}}{=} Yx, \tag{6.163}$$

and those that use its Cholesky factors C_Y such that

$$C_Y C_Y^{\mathrm{T}} = Y \tag{6.164}$$

use the square-root information state

$$s \stackrel{\mathrm{def}}{=} C_Y x. \tag{6.165}$$

6.6.3.4 Information Filter Implementation The implementation equations for the "straight" information filter (i.e., using Y rather than its Cholesky factors) are shown in Table 6.20. These can be derived from the Kalman filter equations and from the definitions of the information matrix and information state. Note the

TABLE 6.20 Information Filter Equations

Observational update:

$$\hat{d}_k(+) = \hat{d}_k(-) + H_k^{\mathrm{T}} R_k^{-1} z_k$$

$$Y_k(+) = Y_k(-) + H_k^{\mathrm{T}} R_k^{-1} H_k$$

Temporal update:

$$A_k \stackrel{def}{=} \Phi_k^{-\mathrm{T}} Y_k(+) \Phi_k^{-1}$$

$$Y_{k+1}(-) = \{I - A_k G_k [G_k^{\mathrm{T}} A_k G_k + Q_k^{-1}]^{-1} G_k^{\mathrm{T}}\} A_k$$

$$\hat{d}_{k+1}(-) = \{I - A_k G_k [G_k^{\mathrm{T}} A_k G_k + Q_k^{-1}]^{-1} G_k^{\mathrm{T}}\} \Phi_k^{-\mathrm{T}} \hat{d}_k(+)$$

TABLE 6.21 SRIF Using Triangularization

Observational update:

$$\begin{bmatrix} C_{Y_k}(-) & H_k^{\mathrm{T}} C_{R_k^{-1}} \\ \hat{s}_k^{\mathrm{T}}(-) & z_k^{\mathrm{T}} C_{R_k^{-1}} \end{bmatrix} T_{\mathrm{obs}} = \begin{bmatrix} C_{Y_k}(+) & 0 \\ \hat{s}_k^{\mathrm{T}}(+) & \epsilon \end{bmatrix}$$

Temporal update:

$$\begin{bmatrix} C_{Q_k^{-1}} & -G_k \Phi_k^{-\mathrm{T}} C_{Y_k}(+) \\ 0 & \Phi_k^{-\mathrm{T}} C_{Y_k}(+) \\ 0 & \hat{s}_k^{\mathrm{T}}(+) \end{bmatrix} T_{\mathrm{temp}} = \begin{bmatrix} \Theta & 0 \\ \Gamma & C_{Y_{k+1}}(-) \\ \tau^{\mathrm{T}} & \hat{s}_{k+1}^{\mathrm{T}}(-) \end{bmatrix}$$

Note: T_{obs} and T_{temp} are orthogonal matrices (composed of Householder or Givens transformations), which lower triangularize the left-hand-side matrices. The submatrices other than s and C_Y on the right-hand sides are extraneous.

similarities in form between these equations and the Kalman filter equations, with respective observational and temporal equations switched.

6.6.3.5 Square-Root Information Filtering The *square-root information filter* is usually abbreviated as *SRIF*. (The conventional square-root filter is often abbreviated as *SRCF*, which stands for *square-root covariance filter*.) Like the SRCF, the SRIF is more robust against roundoff errors than the "straight" form of the filter.

Historical note: A complete formulation (i.e., including both updates) of the SRIF was developed by Dyer and McReynolds [78], using the square-root least-squares methods (triangularization) developed by Golub [98] and applied to sequential least-squares estimation by Lawson and Hanson [167]. The form developed by Dyer and McReynolds is shown in Table 6.21.

6.7 SUMMARY

Although Kalman filtering has been called "ideally suited to digital computer implementation" [95], the digital computer is not ideally suited to the task. The conventional implementation of the Kalman filter—in terms of covariance matrices—is particularly sensitive to roundoff errors.

Many methods have been developed for decreasing the sensitivity of the Kalman filter to roundoff errors. The most successful approaches use alternative representations for the covariance matrix of estimation uncertainty in terms of symmetric products of triangular factors. These fall into three general classes:

1. *Square-root covariance filters*, which use a decomposition of the covariance matrix of estimation uncertainty as a symmetric product of triangular Cholesky factors:

$$P = CC^{\mathrm{T}}.$$

2. *UD covariance filters*, which use a modified (square-root-free) Cholesky decomposition of the covariance matrix:

$$P = UDU^{\mathrm{T}}.$$

3. *Square root information filters*, which use a symmetric product factorization of the information matrix, P^{-1}.

The alternative Kalman filter implementations use these factors of the covariance matrix (or it inverse) in three types of filter operations:

1. *temporal updates*,
2. *observational updates*, and
3. *combined updates* (temporal and observational).

The basic algorithmic methods used in these alternative Kalman filter implementations fall into four general categories. The first three of these categories of methods are concerned with decomposing matrices into triangular factors and maintaining the triangular form of the factors through all the Kalman filtering operations:

1. *Cholesky decomposition methods*, by which a symmetric positive-definite matrix M can be represented as symmetric products of a triangular matrix C:

$$M = CC^{\mathrm{T}} \text{ or } M = UDU^{\mathrm{T}}.$$

   The Cholesky decomposition algorithms compute C (or U and D), given M.
2. *Triangularization methods*, by which a symmetric product of a general matrix A can be represented as a symmetric product of a triangular matrix C:

$$AA^{\mathrm{T}} = CC^{\mathrm{T}} \text{ or } A\acute{D}A^{\mathrm{T}} = UDU^{\mathrm{T}}.$$

   These methods compute C (or U and D), given A (or A and $\acute{D}$).
3. *Rank 1 modification methods*, by which the sum of a symmetric product of a triangular matrix $\acute{C}$ and scaled symmetric product of a vector (rank 1 matrix) $\mathbf{v}$ can be represented by a symmetric product of a new triangular matrix C:

$$\acute{C}\acute{C}^{\mathrm{T}} + s\mathbf{v}\mathbf{v}^{\mathrm{T}} = CC^{\mathrm{T}} \text{ or } \acute{U}\acute{D}\acute{U}^{\mathrm{T}} + s\mathbf{v}\mathbf{v}^{\mathrm{T}} = UDU^{\mathrm{T}}.$$

These methods compute C (or U and D), given $\acute{C}$ (or $\acute{U}$ and $\acute{D}$), s, and $\mathbf{v}$.

The fourth category of methods includes standard matrix operations (multiplications, inversions, etc.) that have been specialized for triangular matrices.

These implementation methods have succeeded where the conventional Kalman filter implementation has failed.

It would be difficult to overemphasize the importance of good numerical methods in Kalman filtering. Limited to finite precision, computers will always make approximation errors. They are not infallible. One must always take this into account in problem analysis. The effects of roundoff may be thought to be minor, but overlooking them could be a major blunder.

PROBLEMS

6.1 An $n \times n$ Moler matrix M has the elements

$$M_{ij} = \begin{cases} i & \text{if } i = j, \\ \min(i, j) & \text{if } i \neq j. \end{cases}$$

Calculate the 3×3 Moler matrix and its lower triangular Cholesky factor.

6.2 Write a MATLAB script to compute and print out the $n \times n$ Moler matrices and their lower triangular Cholesky factors for $2 \leq n \leq 20$.

6.3 Show that the condition number of a Cholesky factor C of $P = CC^{\mathrm{T}}$ is the square root of the condition number of P. (*Hint*: Use SVD.)

6.4 Show that, if A and B are $n \times n$ upper triangular matrices, then their product AB is also upper triangular.

6.5 Show that a square, triangular matrix is singular if and only if one of its diagonal terms is zero. (*Hint*: What is the determinant of a triangular matrix?)

6.6 Show that the inverse of an upper (lower) triangular matrix is also an upper (lower) triangular matrix.

6.7 Show that, if the upper triangular Cholesky decomposition algorithm is applied to the matrix product

$$[H \quad z]^{\mathrm{T}}[H \quad z] = \begin{bmatrix} H^{\mathrm{T}}H & H^{\mathrm{T}}z \\ z^{\mathrm{T}}H & z^{\mathrm{T}}z \end{bmatrix}$$

and the upper triangular result is similarly partitioned as $\begin{bmatrix} U & y \\ 0 & \varepsilon \end{bmatrix}$, then the solution $\hat{x}$ to the equation $U\hat{x} = y$ (which can be computed by back substitution) solves the least-squares problem $Hx \approx z$ with root summed square residual $\|H\hat{x} - z\| = \varepsilon$ (Cholesky's method of least squares).

6.8 The singular-value decomposition of a symmetric, nonnegative-definite matrix P is a factorization $P = EDE^{\mathrm{T}}$ such that E is an orthogonal matrix and $D = \mathrm{diag}(d_1, d_2, d_3, \ldots, d_n)$ is a diagonal matrix with nonnegative elements $d_i \geq 0$, $1 \leq i \leq n$. For $D^{1/2} = \mathrm{diag}(d_1^{1/2}, d_2^{1/2}, d_3^{1/2}, \ldots, d_n^{1/2})$, show that the symmetric matrix $C = ED^{1/2}E^{\mathrm{T}}$ is both a Cholesky factor of P and a square root of P.

6.9 Show that the column vectors of the orthogonal matrix E in the singular value decomposition of P (in Problem 6.8) are the characteristic vectors (eigenvectors) of P, and the corresponding diagonal elements of D are the respective characteristic values (eigenvalues). That is, for $1 \leq i \leq n$, if e_i is the ith column of E, show that $Pe_i = d_i e_i$.

6.10 Show that, if $P = EDE^{\mathrm{T}}$ is a singular-value decomposition of P (defined above), then $P = \sum_{i=1}^{n} d_i e_i e_i^{\mathrm{T}}$, where e_i is the ith column vector of E.

6.11 Show that, if C is an $n \times n$ Cholesky factor of P, then, for any orthogonal matrix T, CT is also a Cholesky factor of P.

6.12 Show that $(I - vv^{\mathrm{T}})^2 = (I - vv^{\mathrm{T}})$ if $|v|^2 = 1$ and that $(I - vv^{\mathrm{T}})^2 = I$ if $|v|^2 = 2$.

6.13 Show that the following formula generalizes the Potter observational update to include vector-valued measurements:

$$C(+) = C(-)[I - VM^{-\mathrm{T}}(M + F)^{-1}V^{\mathrm{T}}],$$

where $V = C^{\mathrm{T}}(-)H^{\mathrm{T}}$ and F and M are Cholesky factors of R and $R + V^{\mathrm{T}}V$, respectively.

6.14 Prove the following lemma: If W is an upper triangular $n \times n$ matrix such that

$$WW^{\mathrm{T}} = I - \frac{\mathbf{v}\mathbf{v}^{\mathrm{T}}}{R + |\mathbf{v}|^2},$$

then[14]

$$\sum_{k=m}^{j} W_{ik}W_{mk} = \Delta_{im} - \frac{\mathbf{v}_i\mathbf{v}_m}{R + \sum_{k=1}^{j}\mathbf{v}_k^2} \tag{6.166}$$

for all i, m, j such that $1 \le i \le m \le j \le n$.

6.15 Prove that the Björck-modified Gram–Schmidt algorithm results in a set of mutually orthogonal vectors.

6.16 Suppose that

$$V = \begin{bmatrix} 1 & 1 & 1 \\ \epsilon & 0 & 0 \\ 0 & \epsilon & 0 \\ 0 & 0 & \epsilon \end{bmatrix},$$

where ϵ is so small that $1 + \epsilon^2$ (but not $1 + \epsilon$) rounds to 1 in machine precision. Compute the rounded result of Gram–Schmidt orthogonalization by the conventional and modified methods. Which result is closer to the theoretical value?

6.17 Show that, if A and B are orthogonal matrices, then

$$\begin{bmatrix} A & 0 \\ 0 & B \end{bmatrix}$$

is an orthogonal matrix.

6.18 What is the inverse of the Householder reflection matrix $I - 2vv^{\mathrm{T}}/v^{\mathrm{T}}v$?

6.19 How many Householder transformations are necessary for triangularization of an $n \times q$ matrix when $n < q$? Does this change when $n = q$?

[14]Kronecker's delta (Δ_{ij}) is defined to equal 1 only if its subscripts are equal ($i = j$) and to equal zero otherwise.

6.20 (Continuous temporal update of Cholesky factors.) Show that all differentiable Cholesky factors $C(t)$ of the solution $P(t)$ to the linear dynamic equation $\dot{P} = F(t)P(t) + P(t)F^{\mathrm{T}}(t) + G(t)Q(t)G^{\mathrm{T}}(t)$, where Q is symmetric, are solutions of a nonlinear dynamic equation $\dot{C}(t) = F(t)C(t) + \frac{1}{2}[G(T)Q(t)G^{\mathrm{T}}(t) + A(t)]C^{-\mathrm{T}}(t)$, where $A(t)$ is a skew-symmetric matrix [140].

6.21 Prove that the condition number of the information matrix is equal to the condition number of the corresponding covariance matrix in the case where neither of them is singular. (The condition number is the ratio of the largest characteristic value to the smallest characteristic value.)

6.22 Prove the correctness of the triangularization equation for the observational update of the SRIF. (Hint: Multiply the partitioned matrices on the right by their respective transposes.)

6.23 Prove the correctness of the triangularization equation for the temporal update of the SRIF.

6.24 Prove to yourself that the conventional Kalman filter Riccati equation

$$P(+) = P(-) - P(-)H^{\mathrm{T}}[HP(-)H^{\mathrm{T}} + R]^{-1}HP(-)$$

for the observational update is equivalent to the information form

$$P^{-1}(+) = P^{-1}(-) + H^{\mathrm{T}}R^{-1}H$$

of Peter Swerling. (Hint: Try multiplying the form for $P(+)$ by the form for $P^{-1}(+)$ and see if it equals I, the identity matrix.)

6.25 Show that, if C is a Cholesky factor of P (i.e., $P = CC^{\mathrm{T}}$), then $C^{-\mathrm{T}} = (C^{-1})^{\mathrm{T}}$ is a Cholesky factor of $Y = P^{-1}$, provided that the inverse of C exists. Conversely, the *transposed* inverse of any Cholesky factor of the information matrix Y is a Cholesky factor of the covariance matrix P, provided that the inverse exists.

6.26 Write a MATLAB script to implement Example 4.4 using the Bierman–Thornton UD filter, plotting as a function of time the resulting RMS estimation uncertainty values of $P(+)$ and $P(-)$ and the components of $\overline{K}$. (You can use the scripts `bierman.m` and `thornton.m`, but you will have to compute UDU^{T} and take the square roots of its diagonal values to obtain RMS uncertainties.)

6.27 Write a MATLAB script to implement Example 4.4 using the Potter square-root filter and plotting the same values as in Problem 6.26.

6.28 Find the lower triangular Cholesky factor of: $\begin{bmatrix} 1 & 2 & 4 \\ 2 & 13 & 23 \\ 4 & 23 & 77 \end{bmatrix}$
Is this its only Cholesky factor?

7

NONLINEAR FILTERING

The principal uses of linear filtering theory are for solving nonlinear problems.
—Harold W. Sorenson, in a private conversation

7.1 CHAPTER FOCUS

7.1.1 The Nonlinear Filtering Problem

Nonlinear Models The subject of this chapter is the practical extension of Kalman filtering to problems with state dynamics governed by nonlinear stochastic differential equations or nonlinear state transformations,

$$\dot{x} = f(x, t) + w(t), \text{ or} \tag{7.1}$$

$$x_k = \phi_{k-1}(x_{k-1}) + w_{k-1}, \tag{7.2}$$

or with a nonlinear transformation from state variables to measurement variables,

$$z_k = h_k(x_k) + w_k, \tag{7.3}$$

or both.

General and Special Solutions Problems of this general type have been modeled by the *conditioned Fokker–Planck equation*, which is a partial differential equation

Kalman Filtering: Theory and Practice using MATLAB®, Third Edition. By Mohinder S. Grewal and Angus P. Andrews

model for the evolution over time of the probability distribution of the state variable, including the effects of measurements. There are no known practical solutions, except in the special case with linear dynamics and measurements. That solution is better known as *Kalman filtering*.

Kalman was able to derive a practical solution in terms of means (i.e., the estimates) and covariances, but it depends on linearity of the functions $h(\cdot)$ and $\phi(\cdot)$ or $f(\cdot)$. The resulting Kalman filter operates in terms of just the means and covariances of the probability distributions. When the underlying probability distributions are Gaussian, these means and covariances uniquely determine the probability distributions. If the state transitions and measurements are linear functions of the state variables, the distributions remain Gaussian and the associated transformations of means and covariances in the temporal and observational update equations of the Kalman filter are exact.

Optimality of the Kalman Gain The computation and application of the Kalman gain—given the means and covariances of the state and measurement—remain exact whether the other parts of the problem are linear or not.

Special Solutions for the Nonlinear Case In order to use the Kalman gain and update for nonlinear problems, the general problem becomes one of finding nonlinear solutions for the following:

1. When state propagation between measurements is nonlinear, how does it transform the estimate (mean) $\hat{x}$ and its associated covariance matrix P?
2. When the measurement $z = h(x)$ is a nonlinear function of the state vector, what is the expected measurement $\hat{z}$ and its associated covariance P_{zz}?
3. The cross-covariance between estimation uncertainty and predicted measurement uncertainty is required for computing the Kalman gain as

$$\bar{K} = P_{xz}/(P_{zz} + R), \tag{7.4}$$

where $P_{xz} = PH^T$ and $P_{zz} = HPH^T$ in the linear case. What is the corresponding cross-covariance P_{xz} when the measurement $z = h(x)$ is a nonlinear function of the state vector?

Nonuniqueness of the Nonlinear Solution There remains the problem that, when the state probability distribution is no longer Gaussian, the first two moments of the distribution (mean and covariance[1]) no longer characterize the distribution. When such probability distributions undergo general transformations, the unknown higher order moments of the distribution can change the value of the first two moments in unpredictable ways. Therefore, *nonlinear filtering cannot be implemented exactly by using just the means and covariances*. Knowing just the

[1]Covariance is the second moment about the *mean*, whereas the standard second moment is about the *origin*.

first two moments of the distributions may be sufficient for the Kalman gain implementation but not for the nonlinear transformations of distributions. All the practical nonlinear filtering methods are approximations of one sort or another.

7.1.2 Practical Solutions

The following nonlinear filtering methods are presented in this chapter:

- **Linearization** essentially linearizes the nonlinear model by using partial derivatives:

$$\Phi \approx \frac{\partial \phi}{\partial x} \tag{7.5}$$

$$F \approx \frac{\partial f}{\partial x} \tag{7.6}$$

$$H \approx \frac{\partial h}{\partial x}. \tag{7.7}$$

  These partial derivatives can also be calculated numerically, as

$$\frac{\partial g}{\partial x_i} \approx \frac{g(x + \Delta x_i) - g(x)}{\Delta x_i},$$

  for $g = \phi, f$, or h. For practical reasons, the magnitudes of the perturbations Δx_i are generally in the order of the standard deviation of uncertainty of x_i. This sort of linearization is used primarily for preliminary analysis of *quasi-linear* (i.e., near-linear) estimation problems by solving the Riccati equation to assess how accurately the state vector might be estimated with the given model parameter values.
- **Extended Kalman filtering (EKF)** evaluates the partial derivatives at the estimated state vector value and uses the full nonlinear functions on the estimate itself:

$$\hat{x}_k(-) \approx \phi(\hat{x}_{k-1}(+))$$
$$\hat{z}_k \approx h(\hat{x}_k(-)).$$

  Extended Kalman filtering was the first successful application of the Kalman filter, and it is still commonly used, although its successful applications have primarily been on problems that are almost linear. In **iterated extended Kalman filtering**, the last equation is closer to

$$\hat{z}_k \approx h(\hat{x}_k(+)).$$

- **Sampling methods** use statistical sampling theory to approximate the nonlinear transformations of means and covariances. Given a mean $\hat{x}$, its associated covariance P, and a nonlinear transformation $g(\cdot)$, a sampling method is used

to select samples χ_i, transform them through $g(\cdot)$, and estimate the new mean and covariance from the transformed samples $g(\chi_i)$. These methods take into account the nonlinear effects on the mean and covariance that are neglected in the linearization methods. However, some of these methods require considerably more computation than conventional Kalman filtering.

- **The unscented Kalman filter (UKF)** is a sampling method with about the same computational complexity as extended Kalman filtering, which makes it quite practical.

Currently, practical nonlinear Kalman filtering methods are still being discovered and refined.

7.1.3 Impractical Solutions

We also present some methods which are exact but computationally impractical for real-time implementation. These methods transform probability distributions, not just the means and covariances. They are used mostly for pedagogical purposes, in demonstrating and analyzing errors introduced by practical but nonexact methods.

7.2 QUASILINEAR FILTERING

The Kalman filter has been applied with some success to problems with nonlinear measurements of nonlinear stochastic systems, provided that the model functions are sufficiently smooth. The nonlinear functions in question are the function f of the stochastic dynamic equation model,

$$\frac{d}{dt}x(t) = f(x,t) + w(t), \tag{7.8}$$

or an equivalent functional transformation in discrete time,

$$\mathbf{x}_k = \phi_k(x_{k-1}) + w_{k-1}, \tag{7.9}$$

and/or the function h of the measurement model,

$$z_k = h_k(x_k) + v_k. \tag{7.10}$$

7.2.1 Quasilinearity

Not much in this world is truly linear, but a relatively large fraction of Kalman filtering problems have been solved through linear approximations of one sort or another. Quasilinearization has been used in many ways, on many problems in science and technology. The essential requirement is that the resulting linear approximation errors be tolerable within the expected accuracy of the solution.

Some nonlinear problems for which Kalman filtering has been successful are deemed *quasilinear* in that the variation of the nonlinear functions h and f or ϕ within the expected ranges of uncertainty about the estimated value of $\mathbf{x}$ is predominantly linear. That is,

$$\phi_k(\mathbf{x} + \delta\mathbf{x}) \approx \phi_k(\mathbf{x}) + \delta\mathbf{x} \frac{\partial}{\partial \mathbf{x}} \phi_k(\mathbf{x})|_{\mathbf{x}} \tag{7.11}$$

$$h_k(\mathbf{x} + \delta\mathbf{x}) \approx h_k(\mathbf{x}) + \delta\mathbf{x} \frac{\partial}{\partial \mathbf{x}} h_k(\mathbf{x})|_{\mathbf{x}}. \tag{7.12}$$

This requires that the functions h and f or ϕ be differentiable at x. Applying linearization techniques to get simple approximate solutions to nonlinear estimation problems generally requires that these functions be twice-continuously differentiable [18, 101].

7.2.1.1 Testing for Quasilinearity

Quasilinearity Metrics The essential idea is that, within reasonably expected variations of the state vector from its estimated value (as determined by the covariance of state estimation uncertainty), the mean-squared errors due to linearization should be insignificant compared to those due to model uncertainties. For measurement nonlinearities, the model uncertainties are characterized by the measurement noise covariance R. For dynamic nonlinearities, the model uncertainties are characterized by the dynamic disturbance noise covariance Q.

The range of perturbations under which these conditions need to be met is generally determined by the magnitude of uncertainty in the estimate. The range can be specified in terms of the standard deviations of uncertainty.

The resulting statistical conditions for linearization can be stated in the following manner:

1. For the temporal state transition function $\phi(x)$: for $N \geq 3$, for $N\sigma$ perturbations of x from $\hat{x}$, the linear approximation error is insignificant compared to Q. That is, for $N \geq 3$, for all perturbations δx of $\hat{x}$ such that

$$(\delta x)^T P^{-1} (\delta x) \leq N^2, \tag{7.13}$$

$$\varepsilon = \underbrace{\phi(\hat{x} + \delta x) - \left[\phi(\hat{x}) + \frac{\partial \phi}{\partial x}\bigg|_{\hat{x}} \delta x\right]}_{\text{approximation error}}, \tag{7.14}$$

$$\varepsilon^T Q^{-1} \varepsilon \ll 1. \tag{7.15}$$

That is, for the expected range of variation of estimation errors, the nonlinear approximation errors are dominated by modeled dynamic uncertainty.

2. For the measurement/sensor transformation $h(x)$: for $N\sigma \geq 3\sigma$ perturbations of $\hat{x}$, the linear approximation error is insignificant compared to R. That is, for some $N \geq 3$, for all perturbations δx of $\hat{x}$ such that

$$(\delta x)^{\mathrm{T}} P^{-1} (\delta x) \leq N^2, \tag{7.16}$$

$$\varepsilon_h = \underbrace{h(\hat{x} + \delta x) - \left[h(\hat{x}) + \left. \frac{\partial h}{\partial x} \right|_{\hat{x}} \delta x \right]}_{\text{approximation error}}, \tag{7.17}$$

$$\varepsilon_h^T R^{-1} \varepsilon_h \ll 1. \tag{7.18}$$

The value of estimation uncertainty covariance P used in Equation 7.16 would ordinarily be the *a priori* value, calculated before the measurement is used. If the measurement update uses what is called the *iterated extended Kalman filter* (IEKF), however, the *a posteriori* value can be used.

Verifying these conditions requires simulating a nominal trajectory for $x(t)$, implementing the Riccati equation to compute the covariance P, sampling the estimate $\hat{x}$ to satisfy the test conditions, and evaluating the test conditions to verify that the problem is adequately quasilinear. The parameter N is essentially a measure of confidence that the linear approximation will be insignificant in the actual application.

Sampling for Testing As a minimum, the perturbations δx can be calculated along the principal axes on the $N\sigma$ equiprobability hyperellipse of the Gaussian distribution of estimation uncertainty. Figure 7.1 is an illustration of such an equiprobability ellipsoid in three dimensions, with arrows drawn along the principal axes of the ellipsoid. These axes and their associated 1σ values can be calculated in MATLAB by using the singular value decomposition (SVD) of a covariance matrix P:

$$P = U \Lambda U^{\mathrm{T}} \tag{7.19}$$

$$= \sum_{i=1}^{n} u_i \sigma_i^2 u_i^{\mathrm{T}} \tag{7.20}$$

$$\delta x_i = N \sigma_i u_i, 1 \leq i \leq n \tag{7.21}$$

$$\delta x_{2n+i} = -N \sigma_i u_i, 1 \leq i \leq n, \tag{7.22}$$

where the vectors u_i are the columns of the orthogonal matrix U in the SVD of P. The principal standard deviations σ_i are the square roots of the diagonal elements of the matrix Λ in the SVD.

The procedure outlined by Equations 7.21 and 7.22 will yield $2n$ perturbation samples, where n is the dimension of x.

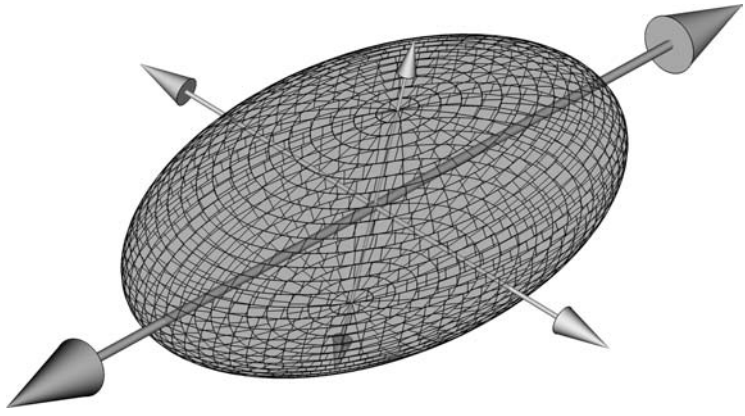


Figure 7.1 Principal axes of equiprobability ellipsoid.

Conditions 7.13 and 7.16 depend on estimation uncertainty. The bigger the uncertainties in $\hat{x}$ are, the larger the perturbations must be for satisfying the quasilinearity constraints.

Example 7.1 (Quasilinearity of Satellite Pseudorange Measurements) Global navigation satellite systems (GNSS) use satellites in orbit around the earth as radio navigation beacons. Receivers on or near the surface of the earth use their internal clocks to measure the time delay Δt between when the signal was broadcast from each satellite in view and when it was received. The product of c, the speed of propagation, times the time delay,

$$\rho = c\,\Delta t$$

is called the *pseudorange* to the satellite from the receiver antenna. The essential geometric model for a single satellite is illustrated in Figure 7.2.

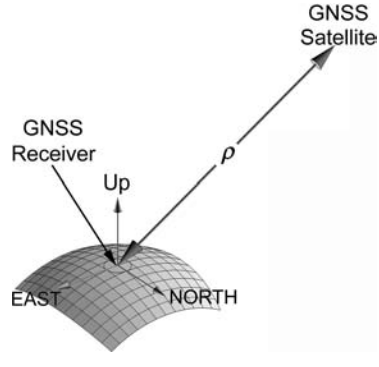


Figure 7.2 GNSS satellite pseudorange ρ.

The GNSS navigation solution for the location of the antenna requires several such pseudoranges to satellites in different directions. Each satellite broadcasts it precise location to the receiver, and the receiver processor is tasked to estimate the location of its antenna from these pseudoranges and satellite positions. This is usually done by extended Kalman filtering, using the derivatives of psuedoranges with respect to receiver location to approximate the measurement sensitivity matrix.

One can use Equation 7.17 to determine whether the pseudorange measurement is truly quasilinear, given the uncertainty in the receiver antenna position, the nonlinear pseudorange measurement model, and the uncertainty in the pseudorange measurement.

For simplicity, we assume that the RMS position uncertainty is uniform in all directions, and

$$P = \begin{bmatrix} \sigma_{\text{pos}}^2 & 0 & 0 \\ 0 & \sigma_{\text{pos}}^2 & 0 \\ 0 & 0 & \sigma_{\text{pos}}^2 \end{bmatrix}, \text{(receiver position covariance)} \tag{7.23}$$

$$R = \sigma_\rho^2 \text{ (pseudorange noise variance)} \tag{7.24}$$

$$\sigma_\rho = 10 \text{ meters (RMS pseudorange noise)} \tag{7.25}$$

$$R_{\text{sat}} = 2.66 \times 10^7 \text{ meters, satellite orbit radius} \tag{7.26}$$

$$R_{\text{rec}} = 6.378 \times 10^6 \text{ meters, receiver radius from earth center} \tag{7.27}$$

$$\alpha = 0^\circ, 30^\circ, 60^\circ, \text{ and } 90^\circ \text{ of elevation of satellite above the horizon} \tag{7.28}$$

$$\rho_0 = \sqrt{R_{\text{sat}}^2 - R_{\text{rec}}^2 \cos(\alpha)^2} - R_{\text{rec}} \sin(\alpha) \text{ (nominal pseudorange)} \tag{7.29}$$

$$X_{\text{sat}} = \rho_0 \begin{bmatrix} -\cos(\alpha) \\ \\ \sin t(\alpha) \end{bmatrix} \text{ (satellite position in east-north-up coordinates)} \tag{7.30}$$

$$\hat{x} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \text{ (estimated receiver position in east-north-up coordinates)} \tag{7.31}$$

$$X_{\text{rec}} = \delta x \tag{7.32}$$

$$= \begin{bmatrix} \delta x_E \\ \delta x_N \\ \delta x_U \end{bmatrix} \text{ (true receiver position in east-north-up coordinates)} \tag{7.33}$$

$$\rho = |X_{\text{rec}} - X_{\text{sat}}| \text{ (true pseudorange)} \tag{7.34}$$

$$= h(\hat{x} + \delta x), \tag{7.35}$$

where Equation 7.34 is the nonlinear formula for pseudorange as a function of perturbations δx of the receiver antenna position in east-north-up coordinates.

The quasilinear approximation for the senstivity of pseudorange to antenna position is

$$H \approx \left.\frac{\partial h}{\partial \delta x}\right|_{\delta x=0} \tag{7.36}$$

$$= [\cos(\alpha) \quad 0 \quad -\sin(\alpha)]. \tag{7.37}$$

In this case, the nonlinear approximation error defined by Equation 7.17 will be a function of the satellite elevation angle α and the perturbation δx.

The RMS nonlinearity error for the six perturbations

$$\delta x = \begin{bmatrix} \pm 3\sigma_{\text{pos}} \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ \pm 3\sigma_{\text{pos}} \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ \pm 3\sigma_{\text{pos}} \end{bmatrix} \tag{7.38}$$

is plotted in Figure 7.3 as a function of σ_{pos} for elevation angles $\alpha = 0^\circ$, 30°, 60°, and 90° above the horizon. The four barely distinguishable solid diagonal lines in the plot are for these four different satellite elevation angles, which have little influence on nonlinearity errors. The horizontal line represents the RMS pseudorange noise, indicating that nonlinear approximation errors are dominated by pseudorange noise for RMS position uncertainties of less than ≈ 10 km.

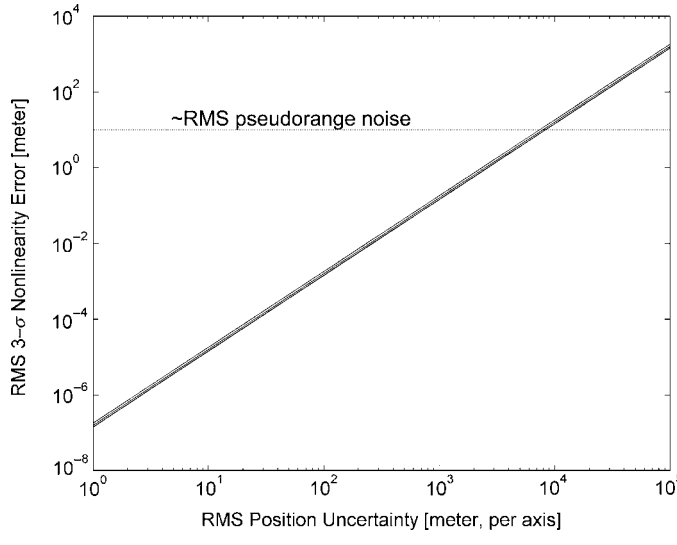


Figure 7.3 Quasilinearity analysis of pseudorange measurement.

Typical RMS position errors in GNSS navigation are on the order of 10 meters, which is well within the quasilinear range. This indicates that extended Kalman filtering is certainly justified for GNSS navigation applications.

7.2.2 Linearization About a Nominal Trajectory

7.2.2.1 Nominal Trajectory A *trajectory* is a particular solution of a stochastic system, that is, with a particular instantiation of the random variates involved. The trajectory is a vector-valued sequence $\{x_k | k = 0,\ 1,\ 2,\ 3, \ldots\}$ for discrete-time systems and a vector-valued function $x(t)$, $0 \leq t$ for continuous-time systems.

The term *nominal* in this case refers to that trajectory obtained when the random variates assume their expected values. For example, the sequence $\{x_k^{\text{nom}}\}$ obtained as a solution of the equation

$$x_k^{\text{nom}} = \phi_{k-1}(x_{k-1}^{\text{nom}}) \tag{7.39}$$

with zero process noise and with the mean x_0^{nom} as the initial condition would be a nominal trajectory for a discrete-time system.

7.2.2.2 Perturbations About a Nominal Trajectory The word *perturbation* has been used by astronomers to describe a minor change in the trajectory of a planet (or any free-falling body) due to secondary forces, such as those produced by other gravitational bodies. Astronomers learned long ago that the actual trajectory can be accurately modeled as the sum of the solution of the two-body problem (which is available in closed form) and a linear dynamic model for the perturbations due to the secondary forces. This technique also works well for many other nonlinear problems, including the problem at hand. In this case, the perturbations are due to the presence of random process noise and errors in the assumed initial conditions.

If the function ϕ in the previous example is continuous, then the state vector x_k at any instant on the trajectory will vary smoothly with small perturbations of the state vector x_{k-1} at the previous instant. These perturbations are the result of *off-nominal* (i.e., off-mean) values of the random variates involved. These random variates include the initial value of the state vector (x_0), the process noise (w_k), and (in the case of the estimated trajectory) the measurement noise (v_k).

If ϕ is continuously differentiable infinitely often, then the influence of the perturbations on the trajectory can be represented by a Taylor series expansion about the nominal trajectory. The likely magnitudes of the perturbations are determined by the variances of the variates involved. If these perturbations are sufficiently small relative to the higher order coefficients of the expansion, then one can obtain a good approximation by ignoring terms beyond some order. (However, one must usually evaluate the magnitudes of the higher order coefficients before making such an assumption.)

Let the symbol δ denote *perturbations* from the nominal,

$$\delta x_k = x_k - x_k^{\text{nom}},$$
$$\delta z_k = z_k - h(x_k^{\text{nom}}, k),$$

so that the Taylor series expansion of $\phi\,(x, k-1)$ with respect to x at $x = x_{k-1}^{\text{nom}}$ is

$$x_k = \phi(x_{k-1}, k-1) \tag{7.40}$$

$$= \phi(x_{k-1}^{\text{nom}}, k-1) + \left.\frac{\partial \phi(x, k-1)}{\partial x}\right|_{x=x_{k-1}^{\text{nom}}} \delta x_{k-1}$$

$$+ \text{higher order terms} \tag{7.41}$$

$$= x_k^{\text{nom}} + \left.\frac{\partial \phi(x, k-1)}{\partial x}\right|_{x=x_{k-1}^{\text{nom}}} \delta x_{k-1}$$

$$+ \text{higher order terms} \tag{7.42}$$

or

$$\delta x_k = x_k - x_k^{\text{nom}} \tag{7.43}$$

$$= \left.\frac{\partial \phi_{k-1}(x)}{\partial x}\right|_{x=x_{k-1}^{\text{nom}}} \delta x_{k-1}$$

$$+ \text{higher order terms.} \tag{7.44}$$

If the higher order terms in δx can be neglected, then

$$\delta x_k \approx \Phi_{k-1}^{[1]} \delta x_{k-1} + w_{k-1}, \tag{7.45}$$

where the first-order approximation coefficients are given by

$$\Phi_{k-1}^{[1]} = \left.\frac{\partial \phi_{k-1}(x)}{\partial x}\right|_{x=x_{k-1}^{\text{nom}}} \tag{7.46}$$

$$= \left.\begin{bmatrix} \frac{\partial \phi_1}{\partial x_1} & \frac{\partial \phi_1}{\partial x_2} & \frac{\partial \phi_1}{\partial x_3} & \cdots & \frac{\partial \phi_1}{\partial x_n} \\ \frac{\partial \phi_2}{\partial x_1} & \frac{\partial \phi_2}{\partial x_2} & \frac{\partial \phi_2}{\partial x_3} & \cdots & \frac{\partial \phi_2}{\partial x_n} \\ \frac{\partial \phi_3}{\partial x_1} & \frac{\partial \phi_3}{\partial x_2} & \frac{\partial \phi_3}{\partial x_3} & \cdots & \frac{\partial \phi_3}{\partial x_n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \phi_n}{\partial x_1} & \frac{\partial \phi_n}{\partial x_2} & \frac{\partial \phi_n}{\partial x_3} & \cdots & \frac{\partial \phi_n}{\partial x_n} \end{bmatrix}\right|_{x=x_{k-1}^{\text{nom}}}, \tag{7.47}$$

an $n \times n$ constant matrix.

7.2.2.3 Linearization of h about a Nominal Trajectory If h is sufficiently differentiable, then the measurement can be represented by a Taylor series:

$$h_k(x_k) = h(x_k^{\text{nom}}, k) + \frac{\partial h_k(x)}{\partial x}\Big|_{x=x_k^{\text{nom}}} \delta x_k + \text{higher order terms} \tag{7.48}$$

or

$$\delta z_k = \frac{\partial h_k(x)}{\partial x}\Big|_{x=x_k^{\text{nom}}} \delta x_k + \text{higher order terms.} \tag{7.49}$$

If the higher order terms in this expansion can be ignored, then one can represent the perturbation in z_k as

$$\delta z_k = H_k^{[1]} \delta x_k, \tag{7.50}$$

where the first-order variational term is

$$H_k^{[1]} = \frac{\partial h_k(x)}{\partial x}\Big|_{x=x_k^{\text{nom}}} \tag{7.51}$$

$$= \begin{bmatrix} \frac{\partial h_1}{\partial x_1} & \frac{\partial h_1}{\partial x_2} & \frac{\partial h_1}{\partial x_3} & \cdots & \frac{\partial h_1}{\partial x_n} \\ \frac{\partial h_2}{\partial x_1} & \frac{\partial h_2}{\partial x_2} & \frac{\partial h_2}{\partial x_3} & \cdots & \frac{\partial h_2}{\partial x_n} \\ \frac{\partial h_3}{\partial x_1} & \frac{\partial h_3}{\partial x_2} & \frac{\partial h_3}{\partial x_3} & \cdots & \frac{\partial h_3}{\partial x_n} \\ \vdots & \vdots & \vdots & & \vdots \\ \frac{\partial h_\ell}{\partial x_1} & \frac{\partial h_\ell}{\partial x_2} & \frac{\partial h_\ell}{\partial x_3} & \cdots & \frac{\partial h_\ell}{\partial x_n} \end{bmatrix}\Bigg|_{x=x_k^{\text{nom}}}, \tag{7.52}$$

which is an $\ell \times n$ constant matrix.

7.2.2.4 Summary of Perturbation Equations in the Discrete Case From Equations 7.45 and 7.50, the linearized equations about nominal values are

$$\delta x_k = \Phi_{k-1}^{[1]} \delta x_{k-1} + w_{k-1}, \tag{7.53}$$

$$\delta z_k = H_k^{[1]} \delta x_k + v_k. \tag{7.54}$$

If the problem is such that the actual trajectory x_k is sufficiently close to the nominal trajectory x_k^{nom} so that the higher order terms in the expansion can be ignored, then this method transforms the problem into a linear problem.

7.2.2.5 Continuous Case In the continuous case, the corresponding nonlinear differential equations for plant and observation are

$$\dot{x}(t) = f(x(t),\, t) + G(t)w(t), \tag{7.55}$$

$$z(t) = h(x(t),\, t) + v(t), \tag{7.56}$$

with the dimensions of the vector quantities the same as in the discrete case.

Similar to the case of the discrete system, the linearized differential equations can be derived as

$$\delta\dot{x}(t) = \left(\frac{\partial f(x(t),\, t)}{\partial x(t)}\bigg|_{x(t)=x^{\text{nom}}}\right)\delta x(t) + G(t)w(t) \tag{7.57}$$

$$= F^{[1]}\delta x(t) + G(t)w(t), \tag{7.58}$$

$$\delta z(t) = \left(\frac{\partial h(x(t),\, t)}{\partial x(t)}\bigg|_{x(t)=x^{\text{nom}}}\right)\delta x(t) + v(t) \tag{7.59}$$

$$= H^{[1]}\delta x(t) + v(t). \tag{7.60}$$

Equations 7.58 and 7.60 represent linearized continuous model equations. The variables $\delta x(t)$ and $\delta z(t)$ are the perturbations about the nominal values, as in the discrete case.

7.2.3 Extended Kalman Filtering

The problem with linearization about the nominal trajectory is that the deviation of the actual trajectory from the nominal trajectory tends to increase with time. As the deviation increases, the significance of the higher order terms in the Taylor series-expansion of the trajectory also increases.

A potential remedy for the deviation problem is to replace the nominal trajectory with the estimated trajectory, that is, to evaluate the Taylor series expansion about the estimated trajectory. If the problem is sufficiently observable (as evidenced by the covariance of estimation uncertainty), then the deviations between the estimated trajectory (along which the expansion is made) and the actual trajectory will remain sufficiently small that the linearization assumption is valid [101, 114].

The principal drawback to this approach is that it tends to increase the real-time computational burden. Whereas Φ, H, and $\overline{K}$ for linearization about a nominal trajectory may have been precomputed off-line, they must be computed in real time as functions of the estimate for linearization about the estimated trajectory.

7.2.3.1 Matrix Evaluations for Discrete Systems The only modification required is to replace x_{k-1}^{nom} by $\hat{x}_{k-1}$ and x_k^{nom} by $\hat{x}_k$ in the evaluations of partial derivatives.

Now the matrices of partial derivatives become

$$\Phi^{[1]}(\hat{x}, k) = \left.\frac{\partial \phi(x, k)}{\partial x}\right|_{x=\hat{x}_k(-)} \tag{7.61}$$

and

$$H^{[1]}(\hat{x}, k) = \left.\frac{\partial h(x, k)}{\partial x}\right|_{x=\hat{x}_k(-)}. \tag{7.62}$$

7.2.3.2 Matrix Evaluations for Continuous Systems The matrices have the same general form as for linearization about a nominal trajectory, except for the evaluations of the partial derivatives:

$$F^{[1]}(t) = \left.\frac{\partial f(x(t), t)}{\partial x(t)}\right|_{x=\hat{x}(t)} \tag{7.63}$$

and

$$H^{[1]}(t) = \left.\frac{\partial h(x(t), t)}{\partial x(t)}\right|_{x=\hat{x}(t)}. \tag{7.64}$$

7.2.3.3 Summary of Implementation Equations For discrete systems linearized about the estimated state,

$$\delta x_k = \Phi^{[1]}_{k-1}\delta x_{k-1} + w_{k-1}, \tag{7.65}$$

$$\delta z_k = H^{[1]}_k \delta x_k + v_k. \tag{7.66}$$

For continuous systems linearized about the estimated state,

$$\delta \dot{x}(t) = F^{[1]}(t)\delta x(t) + G(t)w(t), \tag{7.67}$$

$$\delta z(t) = H^{[1]}\delta x(t) + v(t). \tag{7.68}$$

7.2.4 Discrete Linearized and Extended Filtering

These two approaches to Kalman filter approximations for nonlinear problems yield decidedly different implementation equations. The linearized filtering approach generally has a more efficient real-time implementation, but it is less robust against nonlinear approximation errors than the extended filtering approach.

The real-time implementation of the linearized version can be made more efficient by precomputing the measurement sensitivities, state transition matrices, and Kalman gains. This off-line computation is not possible for the extended Kalman filter,

because these implementation parameters will be functions of the real-time state estimates.

Nonlinear Approximation Errors The extended Kalman filter generally has better robustness because it uses linear approximation over smaller ranges of state space. The linearized implementation assumes linearity over the range of the trajectory perturbations plus state estimation errors, whereas the extended Kalman filter assumes linearity only over the range of state estimation errors. The expected squared magnitudes of these two ranges can be analyzed by comparing the solutions of the two equations

$$P_{k+1} = \Phi_k^{[1]} P_k \Phi_k^{[1]\mathrm{T}} + Q_k,$$
$$P_{k+1} = \Phi_k^{[1]} \{P_k - P_k H_k^{\mathrm{T}} [H_k P_k H_k^{\mathrm{T}} + R_k]^{-1} H_k P_k\} \Phi_k^{[1]\mathrm{T}} + Q_k.$$

The first of these is the equation for the covariance of trajectory perturbations, and the second is the equation for the *a priori* covariance of state estimation errors. The solution of the second equation provides an idea of the ranges over which the extended Kalman filter uses linear approximation. The sum of the solutions of the two equations provides an idea of the ranges over which the linearized filter assumes linearity. The nonlinear approximation errors can be computed as functions of perturbations (for linearized filtering) or estimation errors (for extended filtering) δx by the formulas

$$\varepsilon_x = f(x + \delta x) - f(x) - \frac{\partial f}{\partial x}\delta x,$$
$$\varepsilon_z = \overline{K}(h(x + \delta x) - h(x) - \frac{\partial h}{\partial x}\delta x),$$

where ε_x is the error in the temporal update of the estimated state variable due to nonlinearity of the dynamics and ε_z is the error in the observational update of the estimated state variable due to nonlinearity of the measurement. As a rule of thumb for practical purposes, the magnitudes of these errors should be dominated by the RMS estimation uncertainties. That is, $|\varepsilon|^2 \ll$ trace P for the ranges of δx expected in implementation.

7.2.4.1 Linearized Kalman Filter The block diagram of Figure 7.4 shows the data flow of the estimator linearized about a nominal trajectory of the state dynamics. Note that the operations within the dashed box have no inputs. These are the computations for the nominal trajectory. Because they have no inputs from the rest of the estimator, they can be precomputed off-line.

The models and implementation equations for the linearized discrete Kalman filter that were derived in Section 7.2.2 are summarized in Table 7.1. Note that the last three equations in this table are identical to those of the standard Kalman filter.

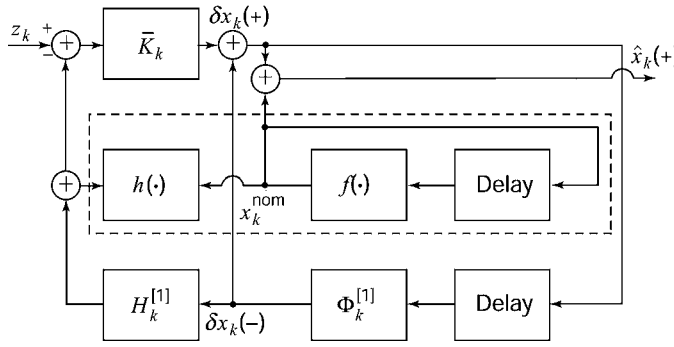


Figure 7.4 Estimator linearized about a nominal state.

7.2.4.2 Discrete Extended Kalman Filter The essential idea of the extended Kalman filter was proposed by Stanley F. Schmidt, and it has been called the *Kalman–Schmidt filter* [27, 238, 240].

The models and implementation equations of the extended Kalman filter that were derived in Section 7.2.3 are summarized in Table 7.2. The last three equations in this table are the same as those for the standard Kalman filter, but the other equations are noticeably different from those of the linearized Kalman filter in Table 7.1.

Example 7.2 (Discrete-Time System)

$$
\begin{aligned}
x_k &= x_{k-1}^2 + w_{k-1}, \\
z_k &= x_k^3 + v_k, \\
Ev_k &= Ew_k = 0, \\
Ev_{k_1} v_{k_2} &= 2\Delta(k_2 - k_1), \\
Ew_{k_1} w_{k_2} &= \Delta(k_2 - k_1), \\
Ex(0) &= \hat{x}_0 = 2, \\
x_k^{\text{nom}} &= 2, \\
P_0(+) &= 1,
\end{aligned}
$$

for which one can use the nominal solution equations from Table 7.1,

$$
\begin{aligned}
\Phi^{[1]}(x_k^{\text{nom}}) &= \frac{\partial}{\partial x}[x^2]\big|_{x=x^{\text{nom}}} \\
&= 4, \\
H^{[1]}(x_k^{\text{nom}}) &= \frac{\partial}{\partial x}(x^3)\big|_{x=x^{\text{nom}}} \\
&= 12,
\end{aligned}
$$

TABLE 7.1 Discrete Linearized Kalman Filter Equations

Nonlinear Nominal Trajectory Model:

$$x_k^{\text{nom}} = \phi_{k-1}(x_{k-1}^{\text{nom}})$$

Linearized Perturbed Trajectory Model:

$$\delta x \overset{\text{def}}{=} x - x^{\text{nom}}$$

$$\delta x_k \approx \frac{\partial \phi_{k-1}}{\partial x}\Big|_{x=x_{k-1}^{\text{nom}}} \delta x_{k-1} + w_{k-1}$$

$$w_k \sim \mathcal{N}(0,\, Q_k)$$

Nonlinear Measurement Model:

$$z_k = h_k(x_k) + v_k, \qquad v_k \sim \mathcal{N}(0, R_k)$$

Linearized Approximation Equations

Linear perturbation prediction:

$$\hat{\delta x}_k(-) = \Phi_{k-1}^{[1]} \hat{\delta x}_{k-1}(+), \qquad \Phi_{(k-1)}^{[1]} \approx \frac{\partial f_{k-1}}{\partial x}\Big|_{x=x_{k-1}^{\text{nom}}}$$

Conditioning the predicted perturbation on the measurement:

$$\hat{\delta x}_k(+) = \hat{\delta x}_k(-) + \overline{K}_k[z_k - h_k(x_k^{\text{nom}}) - H_k^{[1]} \hat{\delta x}_k(-)]$$

$$H_k^{[1]} \approx \frac{\partial h_k}{\partial x}\Big|_{x=x_k^{\text{nom}}}$$

Computing the *a priori* covariance matrix:

$$P_k(-) = \Phi_{k-1}^{[1]} P_{k-1}(+) \Phi_{k-1}^{[1]\text{T}} + Q_{k-1}$$

Computing the Kalman gain:

$$\overline{K}_k = P_k(-) H_k^{[1]\text{T}} [H_k^{[1]} P_k(-) H_k^{[1]\text{T}} + R_k]^{-1}$$

Computing the *a posteriori* covariance matrix:

$$P_k(+) = \{I - \overline{K}_k H_k^{[1]}\} P_k(-)$$

to obtain the discrete linearized filter equations

$$\hat{x}_k(+) = \hat{\delta x}_k(+) + 2,$$

$$\hat{\delta x}_k(+) = 4\hat{\delta x}_{k-1}(+) + \overline{K}_k[z_k - 8 - 48\hat{\delta x}_{k-1}(+)],$$

$$P_k(-) = 16P_{k-1}(+) + 1,$$

$$P_k(+) = [1 - 12\overline{K}_k] P_k(-),$$

$$\overline{K}_k = \frac{12P_k(-)}{144P_k(-) + 2}.$$

TABLE 7.2 Discrete Extended Kalman Filter Equations

Nonlinear Dynamic Model

$$x_k = \phi_{k-1}(x_{k-1}) + w_{k-1}, \quad w_k \sim \mathcal{N}(0,\, Q_k)$$

Nonlinear Measurement Model

$$z_k = h_k(x_k) + v_k, \quad v_k \sim \mathcal{N}(0,R_k)$$

Nonlinear Implementation Equations

Computing the predicted state estimate:

$$\hat{x}_k(-) = \phi_{k-1}(\hat{x}_{k-1}(+))$$

Computing the predicted measurement:

$$\hat{z}_k = h_k(\hat{x}_k(-))$$

Linear approximation:

$$\Phi_{k-1}^{[1]} \approx \frac{\partial \phi_k}{\partial x}\Big|_{x=\hat{x}_{k-1}(-)}$$

Conditioning the predicted estimate on the measurement:

$$\hat{x}_k(+) = \hat{x}_k(-) + \overline{K}_k(z_k - \hat{z}_k), \quad H_{k-1}^{[1]} \approx \frac{\partial h_k}{\partial x}\Big|_{x=\hat{x}_k(-)}$$

Computing the *a priori* covariance matrix:

$$P_k(-) = \Phi_{k-1}^{[1]} P_{k-1}(+)\Phi_{k-1}^{[1]\mathrm{T}} + Q_{k-1}$$

Computing the Kalman gain:

$$\overline{K}_k = P_k(-)H_k^{[1]\mathrm{T}}\left[H_K^{[1]}P_k(-)H_k^{[1]\mathrm{T}} + R_k\right]^{-1}$$

Computing the *a posteriori* covariance matrix:

$$P_k(+) = \{I - \overline{K}_k H_k^{[1]}\}P_k(-)$$

Given the measurements z_k, $k=1, 2, 3$, the values for $P_k(-)$, $\overline{K}_k$, $P_k(+)$, and $\hat{x}(+)$, can be computed. If z_k are not given, then $P_k(-)$, $\overline{K}_k$, and $P_k(+)$ can be computed for covariance analysis results. For large k with large Q, the difference $\hat{x}_k - x_k^{\mathrm{nom}}$ will not stay small, and the results become meaningless.

This situation can be improved by using the extended Kalman filter as discussed in Section 7.2.4.2:

$$\hat{x}_k(+) = \hat{x}_{k-1}^2(+) + \overline{K}_k\{z_k - [\hat{x}_k(-)]^3\},$$
$$P_k(-) = 4[\hat{x}_{k-1}(-)]^2 P_{k-1}(+) + 1,$$

$$\overline{K}_k = \frac{3P_k(-)[\hat{x}_k(-)]^2}{9[\hat{x}_k(-)]^4 P_k(-) + 2},$$
$$P_k(+) = \{1 - 3\overline{K}_k[\hat{x}_k(-)]^2\}P_k(-).$$

These equations are now more complex but should work, provided that Q and R are small.

7.2.5 Continuous Linearized and Extended Filters

The essential equations defining the continuous form of the extended Kalman filter are summarized in Table 7.3. The linearized Kalman filter equations will have x^{nom} in place of $\hat{x}$ as the argument in the evaluations of nonlinear functions and their derivatives.

TABLE 7.3 Continuous EKF Equations

Nonlinear Dynamic Model

$$\dot{x}(t) = f(x(t), t) + w(t) \quad w(t) \sim \mathcal{N}(0, Q(t))$$

Nonlinear Measurement Model

$$z(t) = h(x(t), t) + v(t) \quad v(t) \sim \mathcal{N}(0, R(t))$$

Implementation Equations

Differential equation of the state estimate:

$$\dot{\hat{x}}(t) = f(\hat{x}(t), t) + \overline{K}(t)[z(t) - \hat{x}(t)]$$

Predicted measurement:

$$\hat{z}(t) = h(\hat{x}(t),\ t)$$

Linear approximation equations:

$$F^{[1]}(t) \approx \frac{\partial f(x, t)}{\partial x}\bigg|_{x=\hat{x}(t)}$$
$$H^{[1]}(t) \approx \frac{\partial h(x, t)}{\partial x}\bigg|_{x=\hat{x}(t)}$$

Kalman gain equations:

$$\dot{P}(t) = F^{[1]}(t)P(t) + P(t)F^{[1]\mathrm{T}}(t) + G(t)Q(t)G^{\mathrm{T}}(t) - \overline{K}(t)R(t)\overline{K}^{\mathrm{T}}(t)$$
$$\overline{K}(t) = P(t)H^{[1]\mathrm{T}}(t)R^{-1}(t)$$

7.2.6 Iterated Extended Kalman Filter (IEKF)

Iteration is a long-established approach to solving many nonlinear problems, including Newton's method for solving nonlinear zero-crossing or minimization problems. Several iterated implementation techniques have been used in parts of nonlinear Kalman filtering, including temporal updates and observational updates in the IEKF. We will focus here on the observational update of the covaraince matrix P, because that is where IEKF has been compared favorably with other filtering methods.

7.2.6.1 Iterated Measurement Update A performance comparison of nonlinear filtering methods by Lefebvre et al. [168] showed the IEKF to outperform the EKF, central difference filter (CDF), divided difference filter (DDF), and unscented Kalman filter (UKF) for implementing the measurement update of the covariance matrix. Other studies have ranked sampling-based methods (e.g., UKF) higher overall, but they did not evaluate the observational update independently.

In the EKF, the measurement function $h(t)$ is linearized by partial differentiation,

$$H_k \approx \left.\frac{\partial h}{\partial x}\right|_{x=\hat{x}_k(-)}, \tag{7.69}$$

evaluated at the *a priori* estimate. The IEKF uses successive evaluations of the partial derivatives at better estimates of $\hat{x}$, starting with the *a posteriori* estimate from the extended Kalman filter,

$$\hat{x}_k^{[0]} \stackrel{\text{def}}{=} \hat{x}_{k(+)}, \tag{7.70}$$

which is the zeroth iteration. The final *a posteriori* covariance of estimation uncertainty is calculated only after the last iteration, using the final iterated value of H.

Iteration proceeds for $i = 1, 2, 3, \ldots$ as follows:

$$H_k^{[i]} = \left.\frac{\partial h}{\partial x}\right|_{x=x_k^{[i-1]}} \tag{7.71}$$

$$\overline{K}_k^{[i]} = P_k(-)H_k^{\mathrm{T[i]}}\left[H_k^{[i]}P_k(-)H_k^{[i]\mathrm{T}} + R_k\right]^{-1} \tag{7.72}$$

$$\tilde{z}_k^{[i]} = z_k - h_k(\hat{x}_k^{[i-1]}) \tag{7.73}$$

$$\hat{x}_k^{[i]} = \hat{x}_k(-) + \overline{K}_k^{[i]}\left\{\tilde{z}_k^{[i]} - H_k^{[i]}\left[\hat{x}_k(-) - \hat{x}_k^{[i-1]}\right]\right\} \tag{7.74}$$

until some stopping condition is reached. That stopping condition is often that some vector norm of the difference between successive iterated estimates falls below some

predetermined threshold $\varepsilon_{\text{limit}}$. For example, any of the conditions

$$\left|\hat{x}_k^{[i]} - \hat{x}_k^{[i-1]}\right|_2 < \varepsilon_{\text{limit}} \tag{7.75}$$

$$\left|\hat{x}_k^{[i]} - \hat{x}_k^{[i-1]}\right|_\infty < \varepsilon_{\text{limit}} \tag{7.76}$$

$$(\hat{x}_k^{[i]} - \hat{x}_k^{[i-1]})^{\mathrm{T}} P_k^{-1}(-)(\hat{x}_k^{[i]} - \hat{x}_k^{[i-1]}) < \varepsilon_{\text{limit}} \tag{7.77}$$

might be used as a condition to cease iterating, where the threshold $\varepsilon_{\text{limit}}$ has presumably been chosen based on analysis of real application data.

The final estimate is $\hat{x}_k^{[i]}$, and the associated *a posteriori* covariance matrix of estimation uncertainty is the standard formula:

$$P_k(+) = P_k(-) - \bar{K}_k^{[i]} H_k^{[i]} P_k(-), \tag{7.78}$$

except that the most recently interated value of $H_k^{\{[i]\}}$ is used. In this way, the iteration to obtain a refined estimate of H does not corrupt the statistical record keeping of the Riccati equation.

Example 7.3 (Nonlinear Bearing-Only Measurement Model) As an example of a nonlinear measurement function, consider the sensor geometry illustrated in Figure 7.5, where the angular measurement

$$z_k = \theta_k + v_k \tag{7.79}$$

$$= \arctan\left(\frac{x_k}{d}\right) + v_k \tag{7.80}$$

$$= h(x_k) + v_k \tag{7.81}$$

is the angle θ (plus zero mean white noise $\{v_k\}$), a nonlinear function of the state variable x_k.

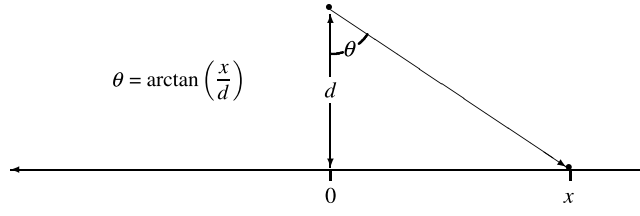


Figure 7.5 Nonlinear measurement model.

If the partial derivative is used as an approximation for the measurement sensitivity matrix,

$$H \approx \left.\frac{\partial h}{\partial x}\right|_x \tag{7.82}$$

$$= \frac{d}{d^2 + x^2}, \tag{7.83}$$

a function of the value of x at which it is evaluated.

Figure 7.6 is a plot of 100 Monte Carlo implementations of the IEKF with five iterations on this problem. The problem parameters used are:

$d = 1$

$R = 10^{-6}$ (1 milliradian RMS sensor noise)

$P = 1$ (mean-squared estimation uncertainty)

$x = 1$ (true value of state variable)

$z = arctan(x) + v \in N(0, \sqrt{R})$ (sample measurement)

$\hat{x} = x + w \in N(0, \sqrt{P})$ (sample estimate)

The plot in Figure 7.6 shows how the 100 randomly sampled initial estimates converge in the IEKF implementation of Equations 7.71 through 7.74 in five iterations. This result was generated by the MATLAB m-file `ExamIEKF.m`. Each time it is run, it will generate 100 independent Monte Carlo simulations with random

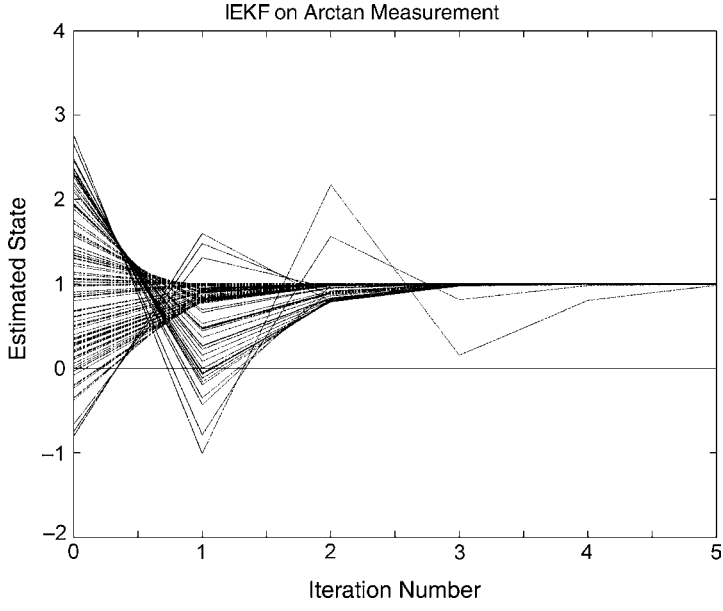


Figure 7.6 One hundred Monte Carlo simulations of IEKF convergence.

estimation errors and random measurement noise. The results shown in Figure 7.6 all converge, which is not usual. You can demonstrate this by running `ExamIEKF.m` many times, in which case some outcomes will demonstrate instances of nonconvergence. This is due to a number of factors, including the unusually large initial covariance of estimation uncertainty and the relatively high degree of nonlinearity of the arctangent function over that range of initial value samples. You can change the value of P in `ExamIEKF.m` to see whether using smaller values of P improves the robustness of the implementation against this sort of divergence.

7.2.7 Higher Order Estimators

The linearized and EKF equations result from truncating a Taylor series expansion of $f(x, t)$ and $h(x, t)$ after the linear terms. Improved model fidelity may be achieved at the expense of an increased computational burden by keeping the second-order terms as well [95, 184, 246].

7.2.8 Biased Errors in Quadratic Measurements

Quadratic dependence of a measurement on the state variables introduces an approximation error ϵ when the *expected value* of the measurement is approximated by the formula $\hat{z} = h(\hat{x}) + \epsilon \approx h(\hat{x})$. It will be shown that the approximation is biased (i.e., $E\langle\epsilon\rangle \neq 0$), and how the expected error $E\langle\epsilon\rangle$ can be calculated and compensated for in the Kalman filter implementation.

7.2.8.1 Quadratic Measurement Model For the sake of simplicity, we consider the case of a scalar measurement. (The resulting correction can be applied to each component of a measurement vector, however.) Suppose that its dependence on the state vector can be represented in the form

$$z = h(x) \tag{7.84}$$

$$= H_1 x + x^{\mathrm{T}} H_2 x + v, \tag{7.85}$$

where H_1 represents the linear (first-order) dependence of the measurement component on the state vector and H_2 represents the quadratic (second-order) dependence. The matrix H_1 will then be a $1 \times n$-dimensioned array and H_2 will be an $n \times n$-dimensioned array, where n is the dimension of the state vector.

7.2.8.2 Quadratic Error Model If one defines the estimation error as $\tilde{x} = \hat{x} - x$, then the expected measurement

$$\hat{z} = E\langle h(x)\rangle \tag{7.86}$$

$$= E\langle H_1 x + x^{\mathrm{T}} H_2 x\rangle \tag{7.87}$$

$$= E\langle H_1(\hat{x} - \tilde{x}) + (\hat{x} - \tilde{x})^{\mathrm{T}} H_2(\hat{x} - \tilde{x})\rangle \tag{7.88}$$

$$= H_1\hat{x} + \hat{x}^{\mathrm{T}} H_2 \hat{x} + E\langle\tilde{x}^{\mathrm{T}} H_2 \tilde{x}\rangle \tag{7.89}$$

$$= h(\hat{x}) + E\langle \text{trace}[\tilde{x}^{\mathrm{T}} H_2 \tilde{x}] \rangle \tag{7.90}$$

$$= h(\hat{x}) + E\langle \text{trace}[H_2(\tilde{x}\tilde{x}^{\mathrm{T}})] \rangle \tag{7.91}$$

$$= h(\hat{x}) + \text{trace}[H_2 P(-)] \tag{7.92}$$

$$= h(\hat{x}) + \varepsilon, \tag{7.93}$$

$$\varepsilon = \text{trace}[H_2 P(-)], \tag{7.94}$$

$$P(-) = E\langle \tilde{x}\tilde{x}^{\mathrm{T}} \rangle, \tag{7.95}$$

where $P(-)$ is the covariance matrix of *a priori* estimation uncertainty. The quadratic error correction should be added in the EKF implementation.

Example 7.4. (Estimating Sensor Scale Factors) Quadratic measurement functions commonly occur in the calibration of linear sensors for which the scale factor s (the ratio between variations of its output z and variations of its input y) and bias b (the value of the output when the input is zero) are also part of the system state vector, along with the input y itself:

$$x = [\, s \quad b \quad y \,], \tag{7.96}$$

$$z = h(x) + v \tag{7.97}$$

$$= sy + b + v. \tag{7.98}$$

The measurement is proportional to the product of the two states $x_1 = s$ and $x_3 = y$. The quadratic form of the second-order measurement model in this example is

$$H_2 = \frac{1\partial^2}{2\partial x^2} h(x)\Big|_{x=0} \tag{7.99}$$

$$= \frac{1}{2} \begin{bmatrix} \dfrac{\partial^2 h}{\partial s^2} & \dfrac{\partial^2 h}{\partial s \partial b} & \dfrac{\partial^2 h}{\partial s \partial y} \\ \dfrac{\partial^2 h}{\partial s \partial b} & \dfrac{\partial^2 h}{\partial b^2} & \dfrac{\partial^2 h}{\partial b \partial y} \\ \dfrac{\partial^2 h}{\partial s \partial y} & \dfrac{\partial^2 h}{\partial b \partial y} & \dfrac{\partial^2 h}{\partial y^2} \end{bmatrix}_{s=b=y=0} \tag{7.100}$$

$$= \begin{bmatrix} 0 & 0 & \dfrac{1}{2} \\ 0 & 0 & 0 \\ \dfrac{1}{2} & 0 & 0 \end{bmatrix}, \tag{7.101}$$

and the correct form for the expected measurement is

$$\hat{z} = h(\hat{x}) + \text{trace}\left\{ \begin{bmatrix} 0 & 0 & \dfrac{1}{2} \\ 0 & 0 & 0 \\ \dfrac{1}{2} & 0 & 0 \end{bmatrix} P(-) \right\} \tag{7.102}$$

$$= \hat{s}\hat{y} + \hat{b} + p_{13}(-), \tag{7.103}$$

where $p_{13}(-)$ is the *a priori* covariance between the scale factor uncertainty and the input uncertainty.

7.2.9 Quasilinear Applications of Filters

Example 7.5. (Damping Parameter Estimation) This example uses the linear damped harmonic oscillator model from Example 4.3 in Chapter 4 as a nonlinear model parameter estimation problem. Assume that ζ (damping coefficient) is unknown and is a constant. Therefore, the damping coefficient can be modeled as a state vector and its value is estimated via linearized and extended Kalman estimators.

Let

$$x_3(t) = \zeta \tag{7.104}$$

and

$$\dot{x}_3(t) = 0 \tag{7.105}$$

Then the dynamic equation of Example 4.3 becomes

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \dot{x}_3(t) \end{bmatrix} = \begin{bmatrix} x_2 \\ -\omega^2 x_1 - 2x_2 x_3 \omega \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} w(t) + \begin{bmatrix} 0 \\ 12 \\ 0 \end{bmatrix}. \tag{7.106}$$

The observation equation is

$$z(t) = x_1(t) + v(t). \tag{7.107}$$

One hundred data points were generated with plant noise and measurement noise set equal to zero, $\zeta = 0.1$, w = 5 rad/s, and initial conditions

$$\begin{bmatrix} x_1(0) \\ x_2(0) \\ x_3(0) \end{bmatrix} = \begin{bmatrix} 0\,\text{ft} \\ 0\ \text{ft/s} \\ 0 \end{bmatrix},$$

$$P(0) = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix},$$

$$Q = 4.47\ (\text{ft/s})^2,$$

$$R = 0.01\ \text{ft}^2.$$

The discrete nonlinear plant and linear observation equations for this model are

$$x_1^k = x_1^{k-1} + Tx_2^{k-1}, \tag{7.108}$$

$$x_2^k = -25Tx_1^{k-1} + (1 - 10Tx_3^{k-1})x_2^{k-1} + 12T + Tw_{k-1}, \tag{7.109}$$

$$x_3^k = x_3^{k-1}, \tag{7.110}$$

$$z_k = x_1^k + v_k. \tag{7.111}$$

The relevant equations from Table 7.1 (discrete linearized Kalman filter equations) and Table 7.2 (discrete EKF equations) have been programmed in MATLAB as `exam53.m` on the accompanying CD. T is sampling interval.

Figure 7.7 shows the estimated position, velocity, and damping factor states (note the nonconvergence due to vanishing gain) using the noise-free data generated from simulating the second-order equation (the same data as in Example 4.3). Figure 7.8 shows the corresponding RMS uncertainties from the EKF. (See Appendix A for descriptions of `exam53.m` and modified versions.)

For the noise-free data, the linearized and EKF results are very close. But for noisy data, convergence for the discrete linearized results is not as fast as convergence for the EKF. Results are a little better with the EKF [176, 185, 189, 222, 238].

Example 7.6 (INS Model) Inertial navigation systems maintain a computational reference frame, which is a set of orthogonal reference axes defined with respect to the inertial sensors (gyroscopes and accelerometers). The attitude error of an inertial navigation system is a set of rotations about these axes representing the rotations between where the system thinks these axes are and where they really are.

Gyroscope Filter. This error can be represented by a model of the form

$$\dot{\Psi} = \Psi \otimes \omega + \varepsilon, \tag{7.112}$$

where

$\Psi = 3 \times 1$ vector containing the attitude alignment errors between the sensor axes frame and the computational reference frame

$\otimes =$ vector cross-product operator

$\omega = 3 \times 1$ vector of the platform inertial angular rate from the trajectory generator

$\varepsilon = 3 \times 1$ vector of composite gyroscope drift rates (algebraic sum of all error sources)

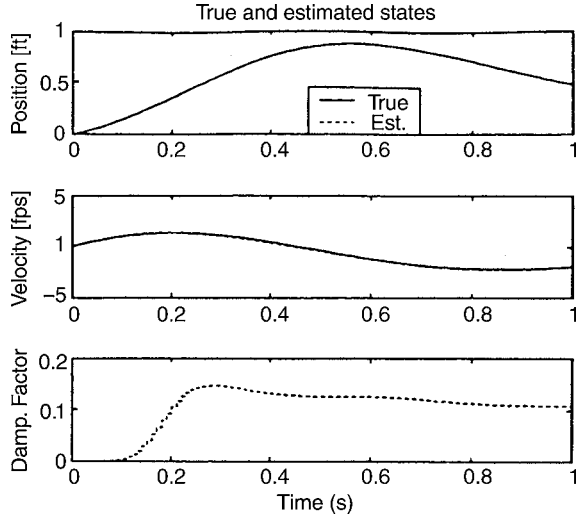


Figure 7.7 State variable estimated by an EKF.

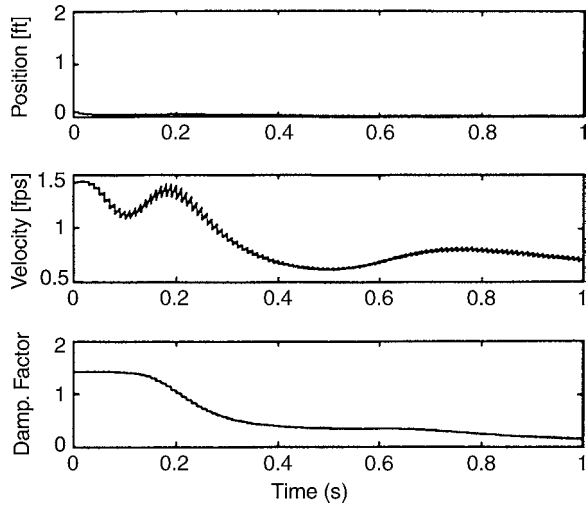


Figure 7.8 RMS uncertainties in estimates.

This attitude error model can be augmented by a 48-state model of the gyroscope parameters (pp. 98–99) as random walks and random ramps. The first-order vector differential equation in a state-space form for the augmented 51×1 state vector is

$$\dot{x}^g(t) = F(t)x^g(t) + w_1(t), \tag{7.113}$$

where the 51-component state vector $x^g(t)$ is composed of the nonredundant components of the following arrays:

Symbol	Ψ	b_g	h_g^1	U_g^1	K_g^1	T_g	b_{gt}	U_{gt}^1
Dimension	3×1	3×1	3×3	3×3	3×6	3×1	3×1	3×6
Subvector	3×1	3×1	9×1	9×1	15×1	3×1	3×1	6×1
$x-$ Indices	$1-3$	$4-6$	$7-15$	$16-24$	$25-39$	$40-42$	$43-45$	$46-51$

(7.114)

The symbol at the top is the array name, with its dimensions shown below it, and the bottom dimension refers to the dimension of the corresponding subvector of its nonredundant terms in the system state vector, shown at the bottom. The matrices h_g, U_g, K_g, and U_{gt} are defined as follows:

h_g is a 3×3 matrix containing unknown scale factor (S_{gi}) and linear axes alignment errors (Δ_{ij}) as components ($i, j = 1, 2, 3$):

$$\begin{bmatrix} S_{g1} & \Delta_{12} & \Delta_{13} \\ \Delta_{21} & S_{g2} & \Delta_{23} \\ \Delta_{31} & \Delta_{32} & S_{g3} \end{bmatrix} \tag{7.115}$$

U_g is a 3×3 matrix of unknown gyroscope mass unbalance parameters $d_{k,j}$:

$$\begin{bmatrix} d_{I1} & d_{01} & d_{S1} \\ d_{S2} & d_{I2} & d_{02} \\ d_{03} & d_{S3} & d_{I3} \end{bmatrix} \tag{7.116}$$

K_g is a 3×6 matrix of unknown gyroscope compliance (g-squared) errors (k_{kji}):

$$\begin{bmatrix} k_{II1} & k_{001} & k_{SS1} & k_{I01} & k_{IS1} & k_{S01} \\ k_{SS2} & k_{II2} & k_{002} & k_{IS2} & k_{S02} & k_{I02} \\ k_{003} & k_{SS3} & k_{II3} & k_{S03} & k_{I03} & k_{IS3} \end{bmatrix} \tag{7.117}$$

U_{gt} is a 3×6 matrix of unknown gyroscope mass unbalance trend parameters.

h_g, U_g, k_g, and U_{gt} have been redimensioned row-wise to form column vectors h_g^1, U_g^1, K_g^1, and U_{gt}^1.

b_g is a 3×1 vector of unknown gyroscope fixed-drift rate parameters.

T_g is a 3×1 vector of unknown nonlinear gyroscope torquer scale factor errors, with elements δS_{gi}.

b_{gt} is a 3×1 vector of unknown gyroscope fixed-drift trend parameters. In expanded form,

$$\begin{bmatrix} \dot{\Psi} \\ \dot{b}_g \\ \dot{h}_g^1 \\ \dot{U}_g^1 \\ \dot{K}_g^1 \\ \dot{T}_g \\ \dot{b}_{gt} \\ \dot{U}_{gt}^1 \end{bmatrix} = \begin{bmatrix} F_{11} & F_{12} & F_{13} & F_{14} & F_{15} & F_{16} & F_{17} & F_{18} \\ 0 & 0 & 0 & 0 & 0 & 0 & F_{27} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & F_{48} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \Psi \\ b_g \\ h_g^1 \\ U_g^1 \\ K_g^1 \\ T_g \\ b_{gt} \\ U_{gt}^1 \end{bmatrix} + \begin{bmatrix} w_\Psi(t) \\ w_{bg}(t) \\ w_{hg}(t) \\ w_{ug}(t) \\ w_{kg}(t) \\ w_{tg}(t) \\ w_{bgt}(t) \\ w_{ugt}(t) \end{bmatrix}, \tag{7.118}$$

where

$$w_1^{\mathrm{T}}(t) = [\, w_\Psi^{\mathrm{T}}(t) \quad w_{bg}^{\mathrm{T}}(t) \quad w_{hg}^{\mathrm{T}}(t) \quad w_{ug}^{\mathrm{T}}(t) \quad w_{kg}^{\mathrm{T}}(t) \quad w_{tg}^{\mathrm{T}}(t) \quad w_{bgt}^{\mathrm{T}}(t) \quad w_{ugt}^{\mathrm{T}}(t) \,] \tag{7.119}$$

is a noise vector of unmodeled effects and

$$F_{11} = \begin{bmatrix} 0 & \omega_3 & -\omega_2 \\ -\omega_3 & 0 & \omega_1 \\ \omega_2 & -\omega_1 & 0 \end{bmatrix}, \quad F_{12} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{7.120}$$

$$F_{13} = \left[\begin{array}{ccc|ccc|ccc} \omega_1 & \omega_2 & \omega_3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \omega_1 & \omega_2 & \omega_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \omega_1 & \omega_2 & \omega_3 \end{array}\right], \tag{7.121}$$

$$F_{14} = \left[\begin{array}{ccc|ccc|ccc} \beta_1 & \beta_2 & \beta_3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \beta_1 & \beta_2 & \beta_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \beta_1 & \beta_2 & \beta_3 \end{array}\right], \tag{7.122}$$

$$F_{15} = \left[\begin{array}{ccccc|ccccc} \beta_{11} & \beta_{33} & \beta_{12} & \beta_{13} & \beta_{23} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \beta_{11} & \beta_{22} & \beta_{12} & \beta_{13} & \beta_{23} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right.$$
$$\left.\begin{array}{ccccc} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \beta_{22} & \beta_{33} & \beta_{12} & \beta_{13} & \beta_{23} \end{array}\right], \tag{7.123}$$

$$\beta_{ij} \overset{\text{def}}{=} \beta_i \beta_j, \tag{7.124}$$

$$F_{16} = \begin{bmatrix} |\omega_1| & 0 & 0 \\ 0 & |\omega_2| & 0 \\ 0 & 0 & |\omega_3| \end{bmatrix}, \tag{7.125}$$

$$F_{17} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} t, \tag{7.126}$$

$$F_{18} = \begin{bmatrix} \beta_1 & \beta_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & \beta_1 & \beta_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & \beta_2 & \beta_3 \end{bmatrix} t, \tag{7.127}$$

$$F_{27} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{7.128}$$

$$F_{48} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{7.129}$$

β is a 3×1 vector of vertical direction cosines (normalized gravity)

$$\beta = (\beta_1, \beta_2, \beta_3)^{\mathrm{T}}, \tag{7.130}$$

β^1 is a (6×1) vector with products of β_i as components

$$(\beta_1^2, \beta_2^2, \beta_3^2, \beta_1\beta_2, \beta_1\beta_3, \beta_2\beta_3), \tag{7.131}$$

and ω_i and β_i are time dependent. Thus, a different system description matrix is computed in each filter cycle.

The corresponding difference equation of continuous-time Equation 7.113,

$$\dot{x}^g(t) = F(t)x^g(t) + w_1(t), \tag{7.132}$$

is

$$x_j^g = \Phi_{j,j-1}^g x_{j-1}^g + w_{j-1}^g, \tag{7.133}$$

where the gyroscope state transition matrix $\Phi_{j,j-1}^g$ is approximated by the first two terms of the power series expansion of the exponential function,

$$\Phi_{j,j-1}^g = I + Ft + F^2\frac{t^2}{2} + \text{higher order terms} \tag{7.134}$$

and w_j^g is normally distributed white noise with zero mean and covariance Q and accounts for gyroscope modeling and truncation errors,

$$w_j^g \sim N(0,\, Q). \tag{7.135}$$

The scalar t is the filter cycle time.

The gyro observation equation is

$$\begin{matrix} 2\times 1 & & 51\times 1 \\ z_j^g = H^g x_j^g + v_j^g, \end{matrix} \tag{7.136}$$

where

$$\begin{matrix} H^g & = & \begin{bmatrix} \alpha^{\mathrm{T}} \\ \gamma^{\mathrm{T}} \end{bmatrix} & [\Phi_{11}\Phi_{12}\Phi_{13}\Phi_{14}\Phi_{15}\Phi_{16}\Phi_{17}\Phi_{18}] \\ 2\times 51 & & 2\times 3 & 3\times 51 \end{matrix}$$

α is the north direction cosine vector (3×1), γ is the west direction cosine vector (3×1), and Φ_{1m} $(m = 1, 2, 3, 4, 5, 6, 7, 8)$ are the appropriate submatrices of the gyroscope state transition matrix $\Phi^g_{j,j-1}$, and $v_j^g \sim \mathcal{N}(0, R)$, which includes noise and errors from sensors.

Accelerometer filter. The difference equation for the accelerometers is

$$x_j^A = \Phi^A_{j,j-1} x^A_{j-1} + w^A_{j-1}, \tag{7.137}$$

where

$$\Phi^A_{j,j-1} = I. \tag{7.138}$$

The 12×1 accelerometer state vector x_j^A is composed of

$$(x^A)^{\mathrm{T}} = [b_A^{\mathrm{T}},\ S_1,\ \delta_{12},\ S_2,\ \delta_{13},\ \delta_{23},\ S_3,\ (FX1 - FI1)^{\mathrm{T}}] \tag{7.139}$$

where

b_A is a 3×1 vector of unknown accelerometer biases, normalized to the magnitude of gravity;

$FI1$ is a 3×1 unknown acceleration-squared nonlinearity for acceleration along the accelerometer input axis; and

$FX1$ is a 3×1 unknown acceleration-squared nonlinearity for acceleration normal to the accelerometer input axis.

Here, S_1, δ_{12}, S_2, δ_{13}, δ_{23}, and S_3 are scalar elements of matrix h_A.

Twelve unknown parameters are modeled as random walk and $w_j^A \sim N(0, Q)$, and white noise includes accelerometer modeling and truncation errors:

$$h_A = \begin{bmatrix} S_1 & \delta_{12} & \delta_{13} \\ 0 & S_2 & \delta_{23} \\ 0 & 0 & S_3 \end{bmatrix}, \tag{7.140}$$

where

S_i = unknown accelerometer scale factor errors $(i = 1, 2, 3)$

δ_{ij} = unknown accelerometer axis nonorthogonalities (misalignments) $(1_i \cdot 1_j)$

Here, β_m is a three-vector $(\beta_1, \beta_2, \beta_3)^{\mathrm{T}}$ of midpoint components of acceleration in platform coordinates,

$$\beta_m^2 = \begin{bmatrix} \beta_1^2 & 0 & 0 \\ 0 & \beta_2^2 & 0 \\ 0 & 0 & \beta_3^2 \end{bmatrix}. \tag{7.141}$$

The accelerometer observation equation is

$$z_j^A = H^A x_j^A + v_j^A, \tag{7.142}$$

where

$$\begin{aligned} H^A = [&\beta_1, \beta_2, \beta_3, \beta_1^2, \beta_1\beta_2, \beta_1\beta_3, \beta_2^2, \beta_2\beta_3, \beta_3^2, (1-\beta_1^2)\beta_1, \\ &(1-\beta_2^2)\beta_2, (1-\beta_3^2)\beta_3]^{\mathrm{T}} \end{aligned} \tag{7.143}$$

and V_j^A is $\sim \mathcal{N}(0, R)$ and white, including sensor noise. The dimension of the observation vector

$$Z = z^g, z^{A\mathrm{T}} \tag{7.144}$$

is 3×1. The EKF equations of Table 7.2 are applied to Equations 7.133 and 7.136 to obtain gyroscope estimates. The EKF equations are applied to Equations 7.137 and 7.142 to obtain accelerometer estimates.

Typical plots of gyroscope fixed-drift, accelerometer fixed-drift, and scale factor estimates are shown in Figures 7.9, 7.10, and 7.11, respectively. Innovation sequence plots for accelerometers and gyroscopes are shown in Figures 7.12 and 7.13, respectively. The results are completely described in reference [102].

Example 7.7. (Freeway Model) This is an application of a discrete-time EKF to estimate the parameters in a macroscopic freeway traffic model [105].

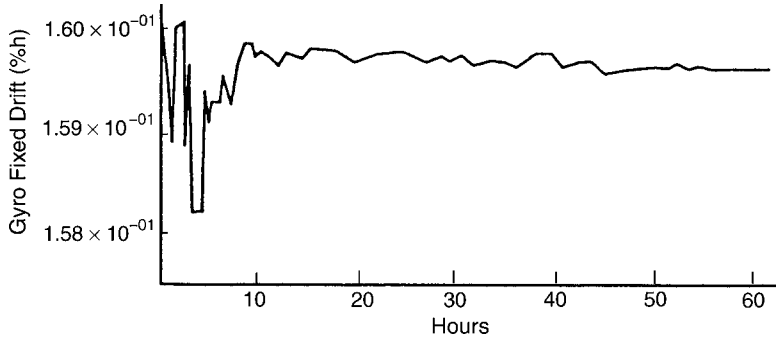


Figure 7.9 Gyro fixed-drift rate estimates.

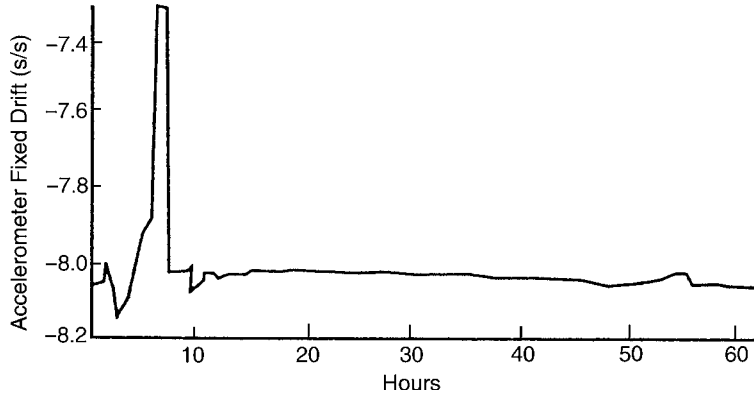


Figure 7.10 Accelerimeter fixed-drift estimates.

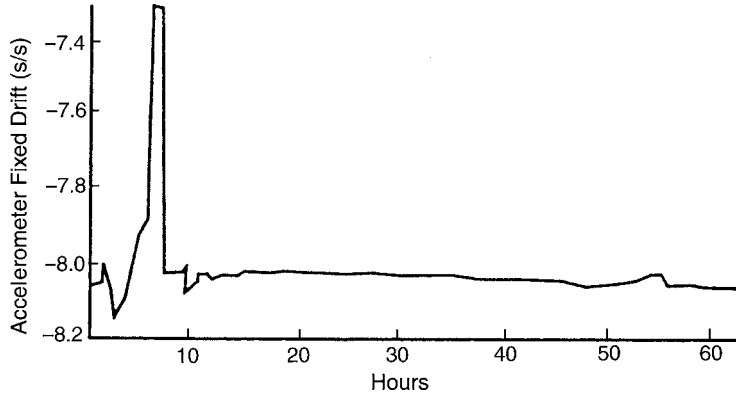


Figure 7.11 Accelerometer scale factor estimates.

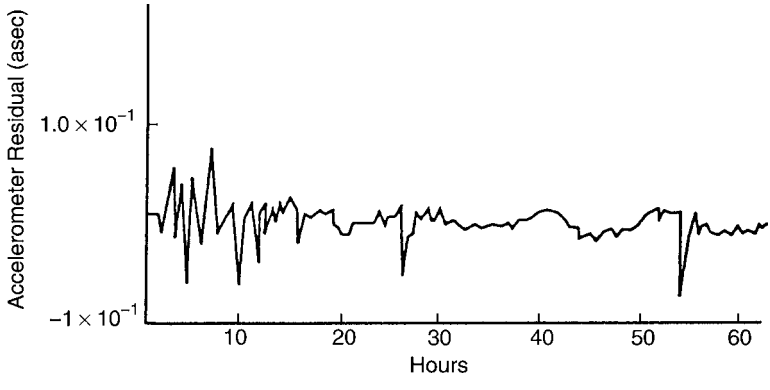


Figure 7.12 Accelerometer residuals.

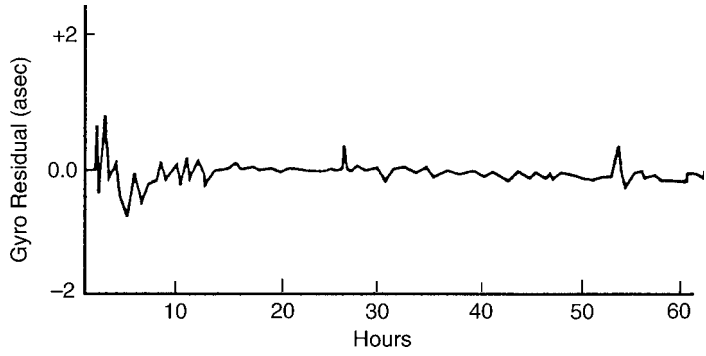


Figure 7.13 Gyroscope residuals.

A dynamic equation is given by

$$u_j^{n+1} = u_j^n + \Delta t\left\{-u_j^n \frac{u_j^n - u_{j-1}^n}{\Delta x} - \frac{1}{T}\left[u_j^n - a - b\rho_j^n + \frac{\nu(\rho_{j+1}^n - \rho_j^n)}{\rho_j^n \Delta x}\right]\right\} + w_j^n,$$

where

$\Delta x_j =$ freeway section length (miles)
$\rho_j^n \equiv \rho(x_j, x_{j+1}; n\Delta t), j = 0, \ldots, N-1$ (vehicles per unit length in the section between x_j and x_{j+1} at time $n\Delta t$)
$u_j^n \equiv u(x_j, x_{j+1}; n\Delta t), j = 0, \ldots, N-1$ (average of speeds of the vehicles in the section between x_j and x_{j+1} at time $n\Delta t$)
$T =$ driver reaction time (unknown)
$\Delta t =$ discrete time interval
$a =$ unknown parameter
$b =$ unknown parameter
$u_e(\rho_j^n) = a + b\rho_j$
$\nu =$ unknown sensitivity factor.

The observation equation is

$$z_j^n = u_j^n + v_j^n.$$

The objective is to estimate the parameters $1/T$, ν, b, and a, where a and b are estimated by least-squares curve filtering to speed and density data, as shown in Figure 7.14. Then the remaining unknown parameters are $1/T$ and v. The technique

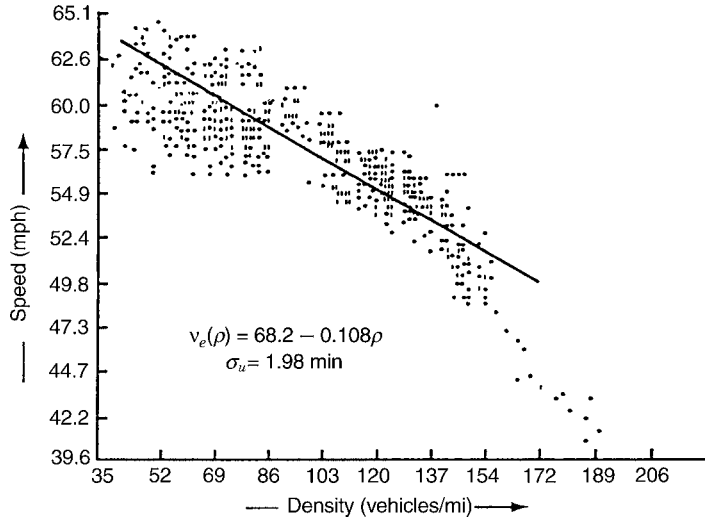


Figure 7.14 Speed–density relationship.

of state augmentation is applied. The parameters $1/T$ and v/T are modeled

$$x_{1j}^n = \frac{1}{T}, \quad x_{2j}^n = \frac{v}{T}.$$

Then

$$x_{1j}^{n+1} = x_{1j}^n, \quad x_{2j}^{n+1} = x_{2j}^n.$$

The nonlinear dynamic equation becomes

$$\begin{bmatrix} u_j^{n+1} \\ x_{1j}^{n+1} \\ x_{2j}^{n+1} \end{bmatrix} = \begin{bmatrix} u_j^n - \dfrac{\Delta t}{\Delta x}(u_j^n)^2 + \dfrac{\Delta t}{\Delta x} u_j^n u_{j-1}^n - \Delta t x_{1j}^n (u_j^n - a - b\rho_j^n) - \dfrac{\Delta t}{\Delta x} x_{2j}^n \dfrac{\rho_{j+1}^n}{\rho_j^n} - 1 \\ x_{1j}^n \\ x_{2j}^n \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} w_j^n.$$

The observation equation is

$$z_j^n = [1\ 0\ 0] \begin{bmatrix} u_j^n \\ x_{1j}^n \\ x_{2j}^n \end{bmatrix} + v_j^n,$$

where w_j^n, v_j^n are zero-mean white Gaussian with covariance Q and R, respectively.

A four-lane, 6000-ft-long section of freeway with no on- or off-ramps and no accidents was simulated on an IBM 360-44 computer. Using a digital simulation of a microscopic model, eight files of data sets (high-flow cases), each containing 20 min of data, section mean speed, and density at 1.5-s intervals were generated.

To demonstrate the application and performance of the methodology of identifying parameters, results from a digital simulation are shown. It has been observed that speed bears a fairly consistent relationship to density. The equilibrium speed–density relationship $u_e(\rho)$ is taken as a least-squares fit by a straight line to the above data and is shown in Figure 7.14. Two parameters, a and b, are estimated by this procedure.

The EKF algorithm is applied to estimate $1/T$ and v/T from the above data. For purposes of numerical computation, it is convenient to define dimensionless variables through the use of nominal values. The nominal values used in the parameter identification algorithm are as follows:

Nominal section mean speed $(u) = 40$ mph,
Nominal value of reaction time $(T) = 30$ s,
Nominal value of sensitivity factor $(v) = 4\ \text{mi}^2/\text{h}$

The time step size $\Delta t = 4.5$ s and the space step size $\Delta x_j = 0.5$ mile are chosen for stability reasons. Figures 7.15 and 7.16 show the typical normalized values of $1/T$ and v/T for the middle (jth) section of a piece of freeway. Actual values of the parameters can be computed by using the above nominal values.

To test the resultant model as a predictor of future traffic conditions, the estimated values of $1/T$, v/T, a, and b are used in the model equation. Section density ρ_j^n and output flow q_{j+1}^{n+1} are computed from the flow. The final model is used to predict the density and speed of the middle section by using the available data from the adjoining

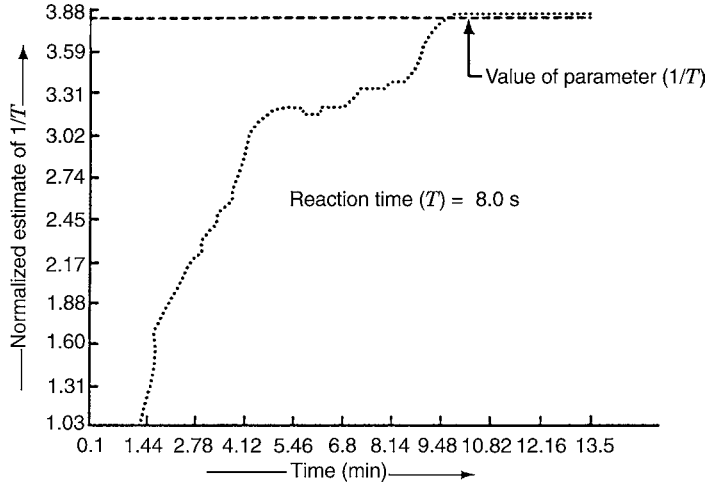


Figure 7.15 Estimation of reaction time.

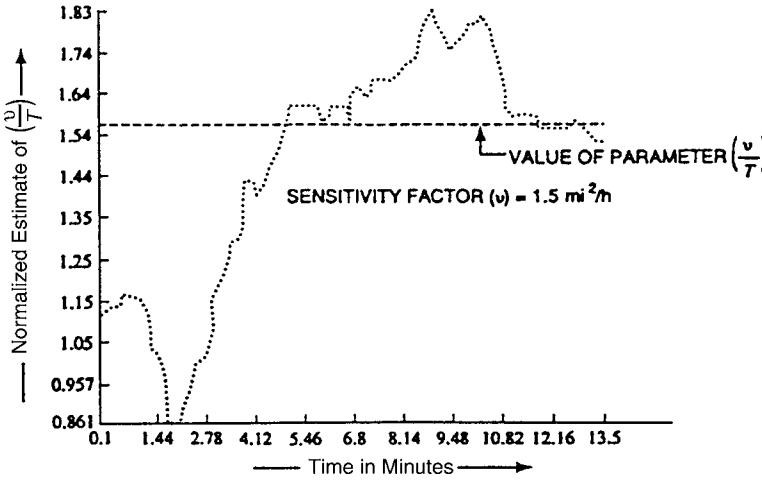


Figure 7.16 Estimation of sensitivity factor.

sections, that is, u_{j-1}^n, ρ_{j-1}^n, ρ_{j+1}^n, and q_j^{n+1}. This model is particularly effective in predicting speed of traffic flow and density in one section of the freeway over 15-min intervals. This time interval is adequate for traffic-responsive control. The single-section density prediction results from the model, and actual density is shown in Figure 7.17. The single-section speed prediction results from the model, and actual section speed is shown in Figure 7.18. These results show that the final model

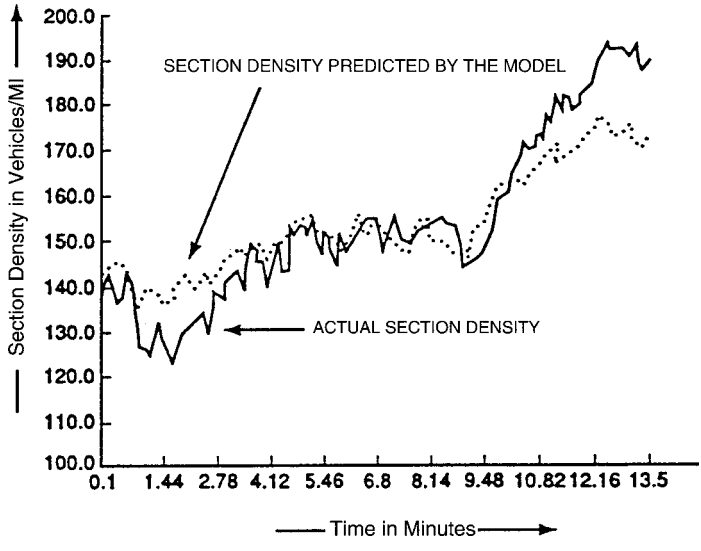


Figure 7.17 Single-section density prediction.

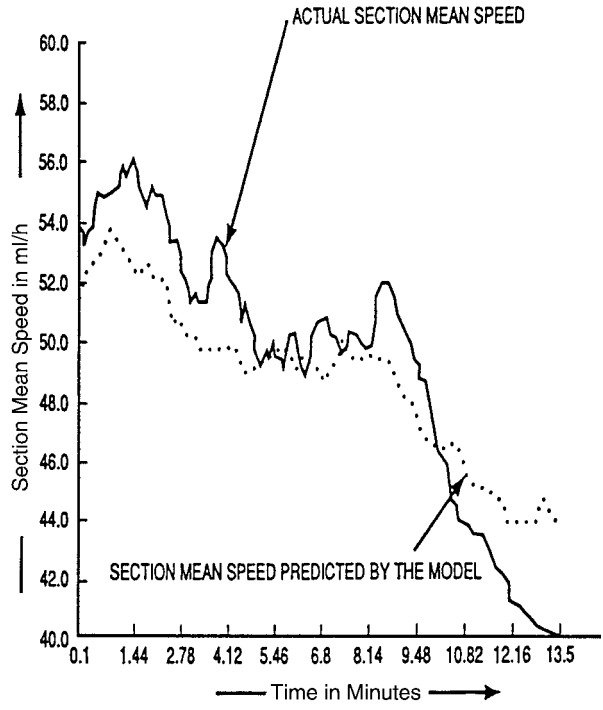


Figure 7.18 Single-section speed prediction.

with the parameter values estimated by the above procedures predicts the traffic conditions (density and speed) satisfactorily.

7.3 SAMPLING METHODS FOR NONLINEAR FILTERING

In problem solving, it is usually better to approximate the solution of the problem at hand than to approximate the real problem by another for which you already have the solution.

Quasilinearization consists essentially of approximating the nonlinear filtering problem by a linear problem and then taking the linear solution.

Sampling methods, on the other hand, recognize the true nonlinear nature of the model and then approximate the solution—including the effects of errors due to nonlinearity.

7.3.1 Historical Background

Sampling Theory The formal study of sampling methods in statistics began about a century ago with the work of Sir Francis Galton (1822–1911) [83]. Since that time,

statisticians have had to contend with many probability distributions, several of which are known only from sampled values. This effort has produced an enormous body of theory and related experience in choosing and using samples to extract statistical information. In the last decade or so, some of this technology has been successfully applied to nonlinear Kalman filtering.

Monte Carlo Analysis In 1946, mathematician Stanislaw Ulam was working on nuclear weapon design at Los Alamos, New Mexico. He had come up with a method for modeling the evolution over time of the distribution of neutrons inside a nuclear device[2] and was describing it to John von Neumann. The concept first came to Ulam when he was playing solitaire while recuperating from an illness [272]. He had wondered what percentage of hands (shuffled decks) are winnable but found the combinatorics too formidable to calculate mentally. But he thought of another approach using random sampling of hands. His colleague Nicholas Metropolis suggested the name *Monte Carlo* (a reference to the gambling casino at Monaco) for this approach [79]. Electronic computers were then being developed and used at Los Alamos, and Monte Carlo analysis was well suited for computer implementation. It soon become the standard method for analyzing general transformations of probability distributions.

Importance Sampling Sample weighting methods are used in to reduce sample size over what would be required using random sampling. In *importance sampling*, samples are chosen by a regularized procedure offering reasonable coverage of the probability domain rather than pseudorandomly according to the probability distribution. Each sample is then weighted according to its *importance*. The importance weighting is often the ratio of local probability density to sampling density, but it may also include sampling and weighting variations to improve the accuracy of estimation for specific statistics—such as means and covariances.

Sigma Points Sigma points χ_i are a structured set of sample points selected by a deterministic procedure based on the known input distribution. The sigma-point selection process is designed to provide reasonable coverage of the input and output probability distributions from the given function ψ and to produce reasonable approximations to the probability distribution of the outputs

$$\xi_i = \psi(\chi_i).$$

In Kalman filtering applications, the only known statistics of these probability distributions are their means (the estimates $\hat{x}$) and their associated covariance matrices of estimation uncertainty, P. A sigma point Kalman filter (SPKF) uses sigma points to approximate transformations of the mean and its covariance matrix.

[2]Ulam is also credited with conceiving the idea of radiation pressure confinement, a critical breakthrough in the development of the hydrogen bomb.

The sigma-point selection process depends on the input mean and covariance of the input probability distribution, and the mean and covariance of the output distribution are calculated using the output transformed samples ξ_i.

Many variants of this approach have been used, some of which are described below.

Central Difference Methods These are sigma-point transformations using numerical first- and second-order numerical derivatives (second central differences) from perturbation samples to select and weight sigma points based on a second-order optimization model.

Unscented Kalman Filter (UKF) About half a century after Ulam's Monte Carlo method was developed, Julier and Uhlmann [122] published an efficient sigma-point sampling method called the *unscented transform* for propagating means and variances through nonlinear transformations. Its application to Kalman filtering is called the *unscented Kalman filter* [123]. The unscented transform selects and weights its samples based on the estimate $\hat{x}$ and Cholesky factors of the covariance matrix P, either the *a posteriori* value (for nonlinear state dynamics) or the *a priori* value (for nonlinear sensors).

The sigma-point weightings used in the unscented transform are designed to provide unbiased estimates of the output mean and covariance, which leaves some degrees of freedom for optimizing other aspects of the transformation. These are used in controlling the first four moments of the approximated output distributions.

The great advantage of the unscented transform for Kalman filtering is its computational efficiency.

Particle Filters *Sequential importance sampling methods* can resample or continue importance sampling to obtain sharper statistical estimates. Like the UKF, *particle filters* use dynamic simulation of these samples as entrained "particles" carried forward in time through nonlinear dynamics, then reconstruct the propagated mean and covariance matrix from the propagated samples. The resulting statistics are used for implementing the Kalman measurement update. With larger sample sizes, particle filters have the potential to obtain better estimates of the means and covariances than the UKF.

Square-Root Sample Methods These are sample-based methods for propagating Cholesky factors of the covariance matrices of the distributions rather than the covariance matrices themselves. These propagation methods are compatible with the more numerically stable implementations of the measurement updates in Kalman filtering.

7.3.2 Higher Order Effects of Nonlinearity

What we usually mean by *nonlinear filtering* is that the deviations of the functions $h(\cdot)$ and $f(\cdot)$ or $\phi(\cdot)$ from linearity are such that linear approximation is not adequate—at least over the ranges of variation expected between the estimate and

its true value and for the allowable tolerances of estimation. Some additional approximation beyond linearity must be done to accommodate it.

7.3.2.1 Sources of Nonlinearity Many applications of Kalman filtering include nonlinearities of one sort or another, including the following:

1. The dynamic model for the application may be inherently nonlinear. This is true, for example, of the first application of Kalman filtering: for determining the trajectories of the Apollo spacecraft on their missions to the moon and back. This nonlinearity can sometimes be ameliorated to some degree by choosing an alternative but mathematically equivalent model to make the resulting dynamics more nearly linear.
2. The measurement model assumes a nonlinear functional form $z = h(x)$. This is fairly common, because most sensor technologies exhibit some degree of input/output nonlinearity. However, it is also becoming fairly common that sensors are integrated with inexpensive microprocessors to compensate for nonlinearities, temperature sensitivities, etc.
3. The list of state variables includes model parameters, such as elements of the state transition matrix Φ, the dynamic coefficient matrix F, the measurement sensitivity matrix H, or the covariances Q and R. Examples of the use of such models include:
   (a) Adaptive Kalman filtering, in which the filter must adapt to slowly changing values of the model parameters. For example, the nonconstant or unknown scale factor of a sensor might be included as a state variable, in which case the output of that sensor can be modeled as the product of two state variables: the nonconstant scale factor and the state variable input to the sensor.
   (b) Hidden Markov modeling applications, in which the objective is to estimate the parameters of the unknown model producing the measurable sensor outputs, and the other state variables of that model are essentially nuisance parameters of the problem. This approach has been used in speech recognition, for example, in which the hidden models represent the articulatory gestures of the human vocal tract, and the measurable variables are spectral and/or ceptstral properties of the vocal tract output (i.e.,speech).

7.3.2.2 Counterintuitive Consequences of Nonlinearity Nonlinear effects can alter your understanding of what *expected value* means.

With Gaussian distributions, the expected value is the mean of the distribution, which is also where the probabiity density function is at its maximum value.

Nonlinearities can alter these relationships.

For example, Figure 7.19 shows what can happen when a Gaussian distribution in polar (r, θ) coordinates is transformed into Cartesian (x, y) coordinates. The mean of the resulting transformed distribution in Cartesian coordinates has a radius quite different from the mean radius in polar coordinates. This plot represents the position

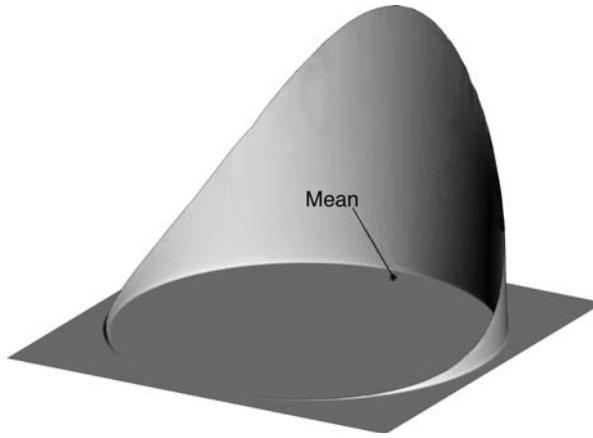


Figure 7.19 Cartesian transformation of a Gaussian distribution in polar coordinates.

probability distribution of a satellite in orbit 200 nautical miles above the earth, with the standard deviation of the in-plane orbit angle equal to 0.5 radian. The mean altitude above the earth in polar coordinates is 200 nautical miles, but the mean location in Cartesian coordinates is actually inside the earth.

Although the true mean of the transformed probability distribution may be inside the earth, it is not where the probability density function (PDF) reaches its maximum value. It is not a good representation of satellite location when we start to look for it. If we expect the satellite to pass underneath us if we wait long enough, we may have to wait a very long time.

7.3.3 The UKF

The UKF[3] was developed by Simon J. Julier and Jeffrey K. Uhlmann in a series of technical papers in the 1990s [122–127].

The approach is based on a sigma-point implementation called the *unscented transform* to propagate means and covariance matrices.

7.3.3.1 The Unscented Transform (UT) The UT is actually a family of transforms, each member of which uses a somewhat different sampling and weighting implementation. This allows some freedom in tailoring the unscented transform to the application, depending on the severity of the nonlinearities and the level of acceptable approximation errors. Still, the number of samples is generally smaller than that of competing sampling-based methods.

[3]The reason for the term *unscented* in this context was not explained in the initial publications.

UT Inputs Inputs to the unscented transform are:

1. The estimate ($\hat{x}$), also called the *mean* of the input distribution.
2. A Cholesky factor C_{xx} of its associated covariance of uncertainty P_{xx}. Possible choices for Cholesky factors C_{xx} of P_{xx} include
   (a) Lower triagular, as computed by the built-in MATLAB function `chol`.
   (b) Upper triangular, as computed by the MATLAB function `utchol` on the CD.
   (c) Symmetric, which can be computed using the built-in MATLAB function for singular value decomposition,

   ```
   [U,Lambda,V] = svd(P),
   ```

   in which case `U*sqrt(Lambda)*U'` is the symmetric positive-definite square root and Cholesky factor of `P`.
   (d) The eigenvector Cholesky factor `CEV = U*sqrt(Lambda)`, the columns of which are scaled eigenvectors of `P`.
3. A state-space transformation function

$$y = g(x),$$

which can be either linear or noninear.

The unspecified vector y is used here because it can have different instantiations in different parts of the Kalman filter.

In applications to Kalman filtering, the function g will be either the state transition function ϕ or the measurement function h, and y will be either a state vector or a measurement vector. When $g = \phi$ is the state transition function, the dimension of y will be n, the dimension of x. When $g = h$, the measurement function, $y = z$ will have dimension l.

The outputs of the UT are $\hat{y}$, the approximated mean of y, and P_{yy}, the approximated covariance (second moment about the mean) of y. The solution includes the effects of nonlinearities, if any. It is exact when the function g is linear and an approximation when g is nonlinear.

The UT selects samples, χ_i of x, depending only on $\hat{x}$ and C_{xx}, and associated weights W_i such that

$$\hat{x} = \sum_i W_i \mathcal{X}_i \tag{7.145}$$

$$P_{xx} = \sum_i W_i (\mathcal{X}_i - \hat{x})(\mathcal{X}_i - \hat{x})^T. \tag{7.146}$$

Each sample χ_i is transformed through the given function g as

$$\xi_i \stackrel{\text{def}}{=} g(\chi_i), \tag{7.147}$$

and the transformed mean $\hat{y}$ and covariance P_{yy} are approximated as

$$\hat{y} \approx \sum_i W_i \xi_i \tag{7.148}$$

$$P_{yy} \approx \sum_i W_i (\xi_i - \hat{y})(\xi_i - \hat{y})^T. \tag{7.149}$$

Different sampling strategies for the UT are summarized in Table 7.4. Derivations of these unscented transforms are generally based on optimizing some performance metrics of the approximation errors, and there is usually enough redundancy built into the model that some free parameters remain for tuning the transform to improve performance on specific applications.

7.3.3.2 The Simplex UT The UT with the smallest sample size is called *simplex*, because it uses vertices of the scaled n-simplex (points x in Euclidean space with components $x_i \geq 0$ such that $\sum_i x_i = 1$) in its sampling strategy. It is designed to minimize the skewness of the solution, which still allows for free parameters that can be tuned to improve performance for specific applications. Its statistical performance may not be as good as that of methods with more samples.

TABLE 7.4 UT Sample Weights

Sampling Strategy	Sample Size*	Sample Values†	Sample Weights	Index Values
Simplex	$n+2$	$\chi_0 = \hat{x}$	$0 \leq W_0 \leq 1$‡	
		$X = [\chi_1\ \chi_2\ \cdots\ \chi_{n+1}]$	$W_1 = 2^{-n}(1 - W_0)$	
		$X = C_{xx}\psi$	$W_2 = W_1$	
		$\Psi = [\psi_1\ \psi_2\ \cdots\ \psi_{n+1}]$	$W_i = 2^{i-1} W_1$	$3 \leq i \leq n+1$
		Algorithms for ψ_i and W_i are in MATLAB function `UTsimplex`		
Symmetric	$2n$	$\chi_i = \hat{x} + \sqrt{n}\, c_i$	$W_i = 1/(2n)$	$1 \leq i \leq n$
		$\chi_{i+n} = \hat{x} - \sqrt{n}\, c_i$	$W_{i+n} = 1/(2n)$	$1 \leq i \leq n$
	$2n+1$	$\chi_0 = \hat{x}$	$W_0 = \kappa/\sqrt{n+\kappa}$	
		$\chi_i = \hat{x} + \sqrt{n+\kappa}\, c_i$	$W_i = 1/[2(n+\kappa)]$	$1 \leq i \leq n$
		$\chi_{i+n} = \hat{x} - \sqrt{n+\kappa}\, c_i$	$W_{i+n} = 1/[2(n+\kappa)]$	$1 \leq i \leq n$
Scaled	$2n+1$	$\chi_0 = \hat{x}$	$W_0^{[\hat{y}]} = \lambda/(n+\lambda)$	
		$\lambda = \alpha^2(n+\kappa) - n$	$W_0^{[P_{yy}]} = W_0^{[\hat{y}]} + (1 - \alpha^2 + \beta)$	
		$\chi_i = \hat{x} + \sqrt{n+\lambda}\, c_i$	$W_i = 1/[2(n+\kappa)]$	$1 \leq i \leq n$
		$\chi_{n+i} = \hat{x} - \sqrt{n+\lambda}\, c_i$	$W_{n+i} = 1/[2(n+\kappa)]$	$1 \leq i \leq n$

*n = dimension of state space.
†c_i = ith column of a Cholesky factor C_{xx} of P_{xx}.
‡W_0 can be varied to tune performance.

The sampling and weighting for this approach are derived in [121] for the alternative state-space vectors

$$\psi \stackrel{\text{def}}{=} C_{xx}^{-1}(x - \hat{x}) \tag{7.150}$$

$$C_{xx}C_{xx}^{T} = P_{xx}, \tag{7.151}$$

for which the mean and covariance

$$\hat{\psi} \stackrel{\text{def}}{=} \mathop{\mathrm{E}}_{x}\langle\psi\rangle \tag{7.152}$$

$$= 0 \tag{7.153}$$

$$P_{\psi\psi} \stackrel{\text{def}}{=} \mathop{\mathrm{E}}_{x}\langle(\psi - \hat{\psi})(\psi - \hat{\psi})^{T}\rangle \tag{7.154}$$

$$= I, \tag{7.155}$$

the identity matrix. However, samples ψ_i can be converted back to x-space by inverting Equation 7.150 as

$$\chi_0 = \hat{x} \tag{7.156}$$

$$\chi_i = \chi_0 + C_{xx}\psi_i, 1 \leq i \leq n + 1 \tag{7.157}$$

The complete algorithm is implemented in the MATLAB function `UTsimplex` on the CD. The function `UTsimplex` includes an optional scaling parameter α for a "scaled" version of the algorithm, with the default value $\alpha = 1$ for the the original (unscaled) simplex UT.

Example 7.8 (Simplex UT with Alternative Choleksy Factors) This is a demonstration of the UT simplex sampling strategy of Table 7.4, using the nonlinear measurement problem with initial mean and covariance

$$\hat{x} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{7.158}$$

$$P_{xx} = \begin{bmatrix} 84 & -64 & -32 & 16 \\ -64 & 84 & 16 & -32 \\ -32 & 16 & 84 & -64 \\ 16 & -32 & -64 & 84 \end{bmatrix}, \tag{7.159}$$

respectively.

All the unscented transforms use Choleksy factors of P_{xx} in sample selection without constraining which form is to be used. The effect of the choice of

Cholesky factor is demonstrated here by using three alternative Cholesky factors of P_{xx}:

$$C_{\mathrm{UT}} = \begin{bmatrix} 3.6956 & -7.4939 & -3.3371 & 1.7457 \\ 0 & 8.3556 & -1.4118 & -3.4915 \\ 0 & 0 & 5.9362 & -6.9830 \\ 0 & 0 & 0 & 9.1652 \end{bmatrix} \text{(upper triangular)} \tag{7.160}$$

$$C_{\mathrm{LT}} = \begin{bmatrix} 9.1652 & 0 & 0 & 0 \\ -6.9830 & 5.9362 & 0 & 0 \\ -3.4915 & -1.4118 & 8.3556 & 0 \\ 1.7457 & -3.3371 & -7.4939 & 3.6956 \end{bmatrix} \text{(lower triangular)} \tag{7.161}$$

$$C_{\mathrm{EV}} = \begin{bmatrix} -7 & -5 & 3 & 1 \\ 7 & 5 & 3 & 1 \\ 7 & -5 & -3 & 1 \\ -7 & 5 & -3 & 1 \end{bmatrix} \text{(eigenvector)}, \tag{7.162}$$

and for the second-order nonlinear measurement function

$$z = h(x) \tag{7.163}$$

$$= \begin{bmatrix} x_2 x_3 \\ x_3 x_1 \\ x_1 x_2 \end{bmatrix} \tag{7.164}$$

For this value of the nonlinear function h and mean $\hat{x} = 0$, the value of $\hat{z}$ for an initial Gaussian distribution can be derived in closed form as

$$\hat{z} = \mathop{\mathrm{E}}_{x} \langle h(x) \rangle \tag{7.165}$$

$$= \frac{1}{(2\pi)^2 \det P_{xx}^{-1}} \int dx_4 \int dx_3 \int dx_2 \int dx_1 h(x) \exp(-x P_{xx}^{-1} x^T/2) \tag{7.166}$$

$$= \begin{bmatrix} P_{2,3}(1 - P_{2,3}^2/P_{2,2}/P_{3,3})^{-3/2} \\ P_{3,1}(1 - P_{3,1}^2/P_{3,3}/P_{1,1})^{-3/2} \\ P_{1,2}(1 - P_{1,2}^2/P_{1,1}/P_{2,2})^{-3/2} \end{bmatrix}. \tag{7.167}$$

That is, the true mean for an initial Gaussian distribution with zero mean depends on the covariance. This dependence is not represented in the EKF.

The results, shown in Table 7.5, were generated by the MATLAB m-file `UTsimplexDemo.m`, using the MATLAB function `UTsimplex`, both of which are on the CD.

The EKF approximation values are also shown in Table 7.5, along with the exact Gaussian solution of Equation 7.167. So that noise covariance does not

TABLE 7.5 Results from Example 7.8: Simplex UT with Alternative Choleksy Factors

Nonlinear Measurement Problem

$$\hat{x} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad P_{xx} = \begin{bmatrix} 84 & -64 & -32 & 16 \\ -64 & 84 & 16 & -32 \\ -32 & 16 & 84 & -64 \\ 16 & -32 & -64 & 84 \end{bmatrix}, \quad z = h(x) = \begin{bmatrix} x_2 x_3 \\ x_3 x_1 \\ x_1 x_2 \end{bmatrix}, \quad R = 0$$

Solution by Simplex UT with $\mathrm{W}_0 = 1/2$

Cholesky Factor	Upper Triangular	Lower Triangular	Eigenvector
Sample P_{zz}	$\begin{bmatrix} 4991 & -4247 & -1231 \\ -4247 & 5313 & 4374 \\ -1231 & 4374 & 14750 \end{bmatrix}$	$\begin{bmatrix} 25775 & -17738 & -29636 \\ -17738 & 13961 & 23304 \\ -29636 & 23304 & 40512 \end{bmatrix}$	$\begin{bmatrix} 4465 & -7093 & -9973 \\ -7093 & 15373 & 6985 \\ -9973 & 6985 & 44381 \end{bmatrix}$
Sample $\hat{z}$	$\begin{bmatrix} -8.38 \\ -19.81 \\ -57.90 \end{bmatrix}$	$\begin{bmatrix} 16 \\ -32 \\ -64 \end{bmatrix}$	$\begin{bmatrix} 15 \\ -33 \\ -55 \end{bmatrix}$

Exact Gaussian Solution

$$\hat{z} = \begin{bmatrix} 16.91 \\ -40.49 \\ -235.55 \end{bmatrix}$$

Solution by EKF Approximation

$$\hat{z} \approx h(\hat{x}) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad H \approx \left.\frac{\partial h}{\partial x}\right|_{x=\hat{x}} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad P_{zz} \approx H P_{xx} H^T = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

mask nonlinear error approximations, we also assume that the sensor noise covariance $R = 0$.

Example 7.8 demonstrates a number of interesting features of the simplex UT for this particular nonlinear problem:

1. The quality of the approximations of $\hat{z}$ and P_{zz} is not what we are used to in EKF of truly quasilinear problems. The initial application of EKF on the Apollo moon missions was able to achieve trajectory estimation accuracies on the order of tens of kilometers at distances on the order of hundreds of thousands of kilometers. However, the degraded performance in this example has more to do with the level of nonlinearity in the application than with qualities of the estimation algorithm. The performance of the EKF approximation in this application is far worse than that of the UT.
2. The sensitivity of the simplex UT to the selection of a Cholesky factor of P_{xx} is troublesome. The simplex sampling strategy is primarily designed to be more efficient than the symmetric sampling strategies—with the potential risk of being less robust against the potential diversity of Cholesky factors.
3. The simplex strategy, although relatively efficient, is not reliably accurate. We must keep in mind, however, that the propagation of means and covariances through nonlinear functions is not uniquely determinable unless the complete initial probability distribution is known. For this reason, all applications of sampling methods to truly nonlinear filtering problems need to be verified through simulation and testing before being trusted in practice.

7.3.3.3 The Symmetric UT This form of the UT is called *symmetric* because its sampling is symmetric about the mean $\hat{x}$. If, for some δx, $\chi_i = \hat{x} + \delta x$ is a sample, then $\chi_{i+n} = \hat{x} - \delta x$ is also a sample.

The more general transform uses the following $2n+1$ samples χ_i and associated weightings W_i:

$$\left.\begin{array}{lll} \chi_0 = \hat{x}, & W_0 = 1/(n+1) & \\ \chi_i = \hat{x} + \sqrt{n+1}c_i, & W_i = 1/[2(n+1)], & 1 \le i \le n \\ \chi_{n+i} = \hat{x} - \sqrt{n+1}c_i, & W_{n+i} = 1/[2(n+1)], & 1 \le i \le n. \end{array}\right\}, \tag{7.168}$$

where the parameter $\kappa = 1$ in Table 7.4. When the mean scaling parameter $\kappa = 0$, the zeroth sigma-point χ_0 can be dropped, leaving only $2n$ samples.

The rest of the symmetric UT has the same implementation equations as the simplex UT, but with different numbers of sigma points. The sigma-point samples χ_i are first transformed as

$$\xi_i = g(\chi_i), \quad 0 \le i \le 2n, \tag{7.169}$$

and the mean and covariance of the transformed samples $g(x)$ are estimated from the weighted transformed samples:

$$\hat{y} = \sum_{i=0}^{2n} W_i \xi_i \tag{7.170}$$

$$P_{yy} = \sum_{i=0}^{2n} W_i (\xi_i - \hat{y})(\xi_i - \hat{y})^T \tag{7.171}$$

7.3.3.4 The Scaled UT This is a further modification of the symmetric UT to allow some sampling and weighting adjustments for improving robustness against higher order nonlinearities beyond second-order.

It uses three adjustable scaling parameters (α, β, κ) to allow for some tuning of the transform for specific applications. The two parameters α and κ can be combined into a single parameter

$$\lambda = \alpha^2 \kappa + (1 - \alpha^2) n \tag{7.172}$$

to simplify the calculation of the samples and weights:

$$\left.\begin{array}{lll} \chi_0 = \hat{x}, & W_0^{[\hat{y}]} = \lambda/(n+\lambda) & \\ & W_0^{[P_{yy}]} = \lambda/(n+\lambda) + 1 - \alpha^2 + \beta & \\ \chi_i = \hat{x} + \sqrt{n+\lambda}\, c_i, & W_i = 1/[2(n+\lambda)], & 1 \le i \le n \\ \chi_{n+i} = \hat{x} - \sqrt{n+\lambda}\, c_i, & W_{n+i} = 1/[2(n+\lambda)], & 1 \le i \le n. \end{array}\right\} \tag{7.173}$$

These samples χ_i are transformed through the function g as

$$\xi_i = g(\chi_i), \quad 0 \le i \le 2n \tag{7.174}$$

to obtain the transformed samples ξ_i.

Two different weighting values $W_0^{[\hat{y}]}$ and $W_0^{[P_{yy}]}$ are used in calculating the transformed mean and covariance:

$$\hat{y} = W_0^{[\hat{y}]} \xi_0 + \sum_{i=1}^{2n} W_i \xi_i \tag{7.175}$$

$$P_{yy} = W_0^{[P_{yy}]} (\xi_0 - \hat{y})(\xi_0 - \hat{y})^T + \sum_{i=1}^{2n} W_i (\xi_i - \hat{y})(\xi_i - \hat{y})^T. \tag{7.176}$$

Scaling Parameters The scaling parameter α is used to shrink the footprint of the samples about the mean and adjust the weightings accordingly. The effect of the scaling parameter on the samples is illustrated in Figure 7.20 for a Gaussian distribution with dimension $n = 2$ and the "eigenvector" Cholesky factor of P_{xx},

$$C_{\mathrm{EV}} = U\sqrt{\Lambda}, \tag{7.177}$$

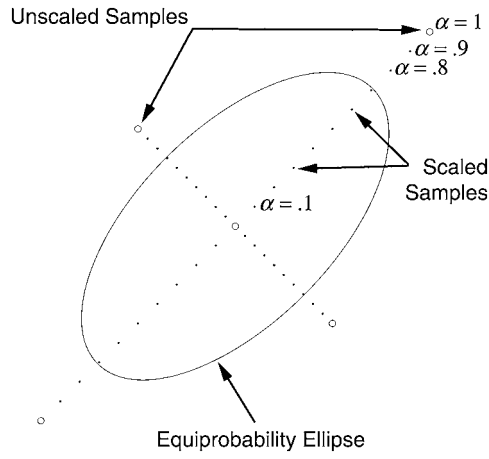


Figure 7.20 α-Scaling of sample points.

where

$$P_{xx} = U\Lambda U^T \tag{7.178}$$

is the singular value decomposition of P_{xx}. In this case, the columns of the Choleksy factor C_{EV} are scaled eigenvectors of P_{xx}, which are the principal axes of the associated equiprobability ellipsoid for Gaussian distributions. When $\alpha = 1$, the transform is the same as the standard UT, with the samples at $\hat{x} \pm \sqrt{n}\sigma_i u_i$. As $\alpha \to 0$, the samples move toward the mean $\hat{x}$ and the influence of nonlinearities beyond second-order in g on the approximated $\hat{y}$ and P_{yy} is theoretically reduced.

The parameter κ is used to apply a different weight on the mean $\hat{x}$ in calculating both $\hat{y}$ and P_{yy}. When $\kappa = 0$, the sample χ_0 is effectively omitted. Some users have tried negative values of κ, even though the positive definiteness of the resulting value of P_{yy} cannot be guaranteed when $\kappa < 0$.

The scaling parameter β is used to create two different weights W_0: one for calculating the mean $\hat{y}$ and another for calculating its covariance P_{yy}.

Example 7.9 (Symmetric UT with Alternative Cholesky Factors) Results from using the symmetric UT sampling strategy with the same nonlinear measurement problem as in Example 7.8 are shown in Table 7.6. These were generated by MATLAB m-file `UTscaledDemo.m` using the MATLAB function `UTscaled`, both of which are on the CD, and should be compared to the results in Table 7.5 for the simplex UT.

These results are not very different from those obtained with the simplex UT and exhibit similar sensitivities to the choice of Cholesky factor. They are vastly better than the EKF results, however.

TABLE 7.6 Results from Example 7.9: Symmetric UT with Alternative Cholesky Factors

Nonlinear Measurement Problem

$$\hat{x} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad P_{xx} = \begin{bmatrix} 84 & -64 & -32 & 16 \\ -64 & 84 & 16 & -32 \\ -32 & 16 & 84 & -64 \\ 16 & -32 & -64 & 84 \end{bmatrix}, \quad z = h(x) = \begin{bmatrix} x_2 x_3 \\ x_3 x_1 \\ x_1 x_2 \end{bmatrix}, \quad R = 0$$

Solution by Scaled UT with $\alpha = 1$, $\beta = 2$, $\kappa = 2$

Cholesky Factor	Upper Triangular	Lower Triangular	Eigenvector
Sample P_{zz}	$\begin{bmatrix} 141 & 332 & -604 \\ 332 & 785 & -1427 \\ -604 & -1427 & 8476 \end{bmatrix}$	$\begin{bmatrix} 1738 & -1829 & -3657 \\ -1829 & 2048 & 4096 \\ -3657 & 4096 & 8192 \end{bmatrix}$	$\begin{bmatrix} 4465 & -7093 & -9973 \\ -7093 & 15373 & 6985 \\ -9973 & 6985 & 44381 \end{bmatrix}$
Sample $\hat{z}$	$\begin{bmatrix} -8.38 \\ -19.81 \\ -57.90 \end{bmatrix}$	$\begin{bmatrix} 16 \\ -32 \\ -64 \end{bmatrix}$	$\begin{bmatrix} 15 \\ -33 \\ -65 \end{bmatrix}$

Exact Gaussian Solution

$$\hat{z} = \begin{bmatrix} 16.91 \\ -40.49 \\ -235.55 \end{bmatrix}$$

Solution by EKF Approximation

$$\hat{z} \approx h(\hat{x}) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad H \approx \left.\frac{\partial h}{\partial x}\right|_{x=\hat{x}} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad P_{zz} \approx H P_{xx} H^T = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

7.3.4 Unscented Kalman Gain

The standard Kalman gain for linear models is

$$\overline{K}_k \stackrel{\text{def}}{=} P_{k,(-)}H_k^T/[H_k P_{k,(-)}H_k^T + R_k]. \tag{7.179}$$

When the model is nonlinear, the equivalent factors $P_{k,(-)}$ and $P_{zz} \equiv H_k P_{k,(-)}H_k^T$ can be approximated using the UT. The only missing factor is the cross-covariance term, which can be approximated from the UT samples and weights as

$$P_{k,(-)}H_k^T \approx \sum_i W_i(\chi_i - \hat{x}_{k,(-)})(\xi_i - \hat{z}_k)^T. \tag{7.180}$$

These partial results from the UT can then be used to approximate the Kalman gain as the matrix fraction

$$\overline{K}_k = \left[\sum_i W_i(\chi_i - \hat{x}_{k,(-)})(\xi_i - \hat{z}_k)^T\right] \Bigg/ \left[R_k + \sum_i W_i(\xi_i - \hat{z}_k)(\xi_i - \hat{z}_k)^T\right]. \tag{7.181}$$

7.3.5 UKF Implementation

The UKF implementation using the UT is summarized in Table 7.7, where "UT" stands for an unscented transform. A MATLAB implementation is contained in the functions `UKFtempUD` (temporal update) and `UKFmeasUD` (measurement update) on the CD.

TABLE 7.7 UKF Implementation*

	Temporal Update
$C_{k-1,(+)}$	$=$ a Cholesky factor of $P_{k-1,(+)}$
$[\hat{x}_{k,(-)}, P_{xx}]$	$= \text{UT}(\hat{x}_{k-1,(+)}, C_{k-1,(+)}, @\phi, \kappa, \alpha, \beta)$
$P_{k,(-)}$	$= P_{xx} + Q_k$
	Measurement Update
$C_{k,(-)}$	$=$ a Cholesky factor of $P_{k,(-)}$
$[\hat{z}_k, P_{zz}]$	$= \text{UT}(\hat{x}_{k,(-)}, C_{k,(-)}, @h, \kappa, \alpha, \beta)$
P_{xz}	$= [\sum_i W_i(\chi_i - \hat{x}_{k,(-)})(\xi_i - \hat{z}_k)^T]$
K_k	$= P_{xz}/[P_{zz} + R_k]$
$\hat{x}_{k,(+)}$	$= \hat{x}_{k,(-)} + K_k(z_k - \hat{z}_k)$
$P_{k,(+)}$	$= P_{k,(-)} - K_k P_{xz}^T$

*The prefix "@" is for passing functions as arguments in MATLAB.

7.4 SUMMARY

7.4.1 Important Points to Remember

Discrete-time and continuous-time estimators derived in Chapter 4 can be applied to nonlinear problems using the following approaches:

1. Linearization of nonlinear plant and observation models about a fixed *nominal trajectory*. See Table 7.1 for the discrete-time linearized Kalman filter using this approach.
2. EKF, which requires linearization of plant and observation equations about the *estimated trajectory* at every time step $[\hat{x}_k(-)]$. See Tables 7.2 and 7.3 for the filter equations.
3. Higher order estimators, which are generally considered too computationally complex for real-time applications.
4. Sampling methods for estimating the means and variances of nonlinear functions of the state variable distributions. The most efficient of these is the UKF, which has about the same computational complexity as EKF.

Table 7.8 summarizes the linear and nonlinear filtering methods covered, pointing out some discriminating differences between alternative approaches. Additional details and options are presented in the chapter.

The only part of the Kalman filter implementation that remains unchanged across all these methods is the application of the Kalman gain in the observational update, including the covariance update as well as the update of the estimate.

Sampling methods such as UKF apply as well to linear models, including mixed models with linear as well as nonlinear parts. The sample weightings for these methods are generally constrained to give the correct answer when the model is linear and the distribution is Gaussian. Sampling methods also include effects of nonlinearities which are not included in the other methods in Table 7.8.

7.4.2 Limitations of Nonlinear Kalman Filters

All nonlinear Kalman filtering methods are approximations of one sort of another. The fundamental problem is that nonlinear transformations of probability distributions are not uniquely determined by means and covariances alone, the way that linear transformations of Gaussian distributions can be characterized.

The remarkable thing is that these approximations work as well as they do.

Example 7.10 (Different Probability Distributions with Identical Means and Variances) This example demonstrates that two distributions of x with identical means and variances, when transformed by the same nonlinear function $z = x^2$, can produce different means and variances of z. In the context of Kalman filtering, the mean of z is the predicted measurement and the variance of z is the equivalent of HPH^{T} (that part of the variance of innovations without the noise).

TABLE 7.8 Summary of Selected Filtering Methods

	Implementation				
	Linear	Nonlinear			
Update	Kalman (KF) & Kalman-Bucy (KBF)	Linearized (LKF)	Extended (EKF)	Iterated (IEKF)	Unscented (UKF)
Temporal Estimate					
Continuous	$\dot{x} = Fx$	$\dot{x} = f(x)$	←(same)	←(same)	UT*
Discrete	$x_k = \Phi_{k-1}x_{k-1}$	$\dot{\Phi} \approx \left.\frac{\partial f}{\partial x}\right\|_{x_{\text{nom}}} \Phi$	$\dot{\Phi} \approx \left.\frac{\partial f}{\partial x}\right\|_{\hat{x}} \Phi$	←(same)	UT*
		$\Phi(t_{k-1}) = \mathrm{I}$	←(same)	←(same)	UT*
Covariance					
Continuous	$\dot{P} = FP + PF^{\mathrm{T}} + Q$	$F \approx \left.\frac{\partial f}{\partial x}\right\|_{x_{\text{nom.}}}$	$F \approx \left.\frac{\partial f}{\partial x}\right\|_{\hat{x}}$	←(same)	UT*
Discrete	$P_k = \Phi_{k-1}P_k\Phi_{k-1}^{\mathrm{T}} + Q_{k-1}$	←(same)	←(same)	←(same)	UT*
Observational					
Measurement	$\hat{z}_k = H_k\hat{x}_k$	$\hat{z}_k = h_k(\hat{x}_k)$	←(same)	←(same)	UT*
Kalman Gain	$\overline{K} = PH^{\mathrm{T}}/(HPH^{\mathrm{T}} + R)$	$H \approx \left.\frac{\partial h}{\partial x}\right\|_{x_{\text{nom}}}$	$H \approx \left.\frac{\partial h}{\partial x}\right\|_{\hat{x}(-)}$	$H \approx \left.\frac{\partial h}{\partial x}\right\|_{\hat{x}(+)}$	UT*
Estimate	$\hat{x}(+) = \hat{x}(-) + \overline{K}_k(z_k - \hat{z}_k)$	←(same)	←(same)	←(same)	←(same)
Covariance	$P_k(+) = P_k(-) - \overline{K}_k H_k P_k(-)$	←(same)	←(same)	←(same)	←(same)

*The UT propagates the means $\hat{x}$ and $\hat{z}$ and covariances (P and HPH^{T}) of the measurements as well as the estimates, and for linear as well as nonlinear models.

The exponential transformation of a Gaussian distribution with mean μ_G and variance σ_G^2 is a lognormal distribution, with mean

$$\mu_{ln} = \exp(\mu_G + \sigma_G^2/2),$$

variance

$$\sigma_{ln}^2 = [\exp(\sigma_G^2) - 1]\exp(2\mu_G + \sigma_G^2),$$

and PDF

$$dp_{ln}(x) = \frac{\exp\left[-\dfrac{(\log(x) - \log(\mu_{\ln}) + \xi/2)^2}{2\xi}\right] dx}{x\sqrt{2\pi\xi}}$$

$$\xi = \log\left(1 + \frac{\sigma_{\ln}{}^2}{\mu_{\ln}{}^2}\right)$$

for $x > 0$.

Figure 7.21 is a plot of a Gaussian distribution and a lognormal distribution, both with mean $\mu = 1$ and variance $\sigma^2 = 1$.

If the nonlinear measurement transformation

$$z = h(x) = x^2,$$

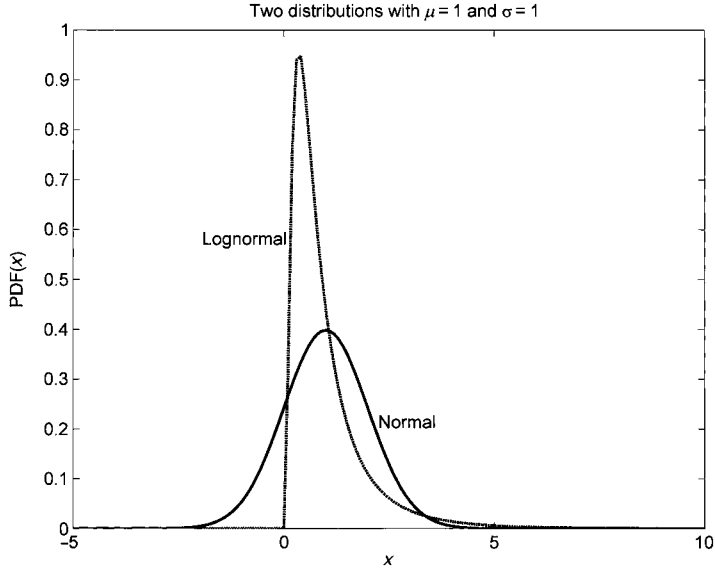


Figure 7.21 Gaussian and lognormal distributions with identical means and variances.

then the PDF of z will be

$$\frac{dp}{dz} = \frac{dp}{dx} \div \frac{dz}{dx}$$
$$= \frac{dp\,1}{dx2x}$$
$$dp(z) = \frac{dp(\sqrt{z})\,dz}{2\sqrt{z}},$$

whatever the probability distribution of x. The corresponding z distributions for the two distributions in Figure 7.21 are plotted in Figure 7.22. The means and variances of all four distributions are given in Table 7.9. In this particular example, the true means after the quadratic transformation are close, but the variances differ by more than an order of magnitude. This demonstrates that one cannot expect to propagate means and covariances accurately through nonlinear transformation without knowing the full initial distributions—not just the initial means and covariances.

This example also provides two cases in which $\hat{z} \neq h(\hat{x})$, because $\hat{x} = 1$ in both cases but $\hat{z} \approx 2 \neq \hat{x}^2$.

7.4.3 Developmental Status of Nonlinear Kalman Filtering

Technological Maturity Recent developments in sampling-based methods for the application of Kalman filtering to nonlinear problems have clearly demonstrated that the technology cannot yet be considered mature in the sense that the advances have gone about as far as they can go.

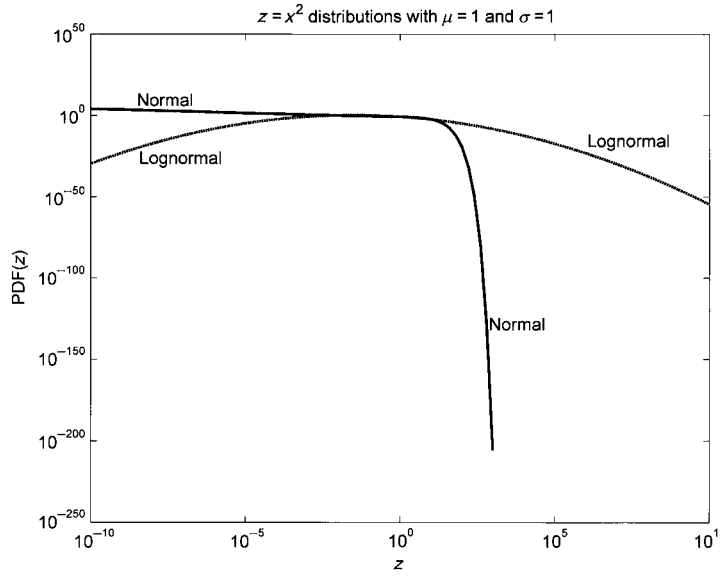


Figure 7.22 The z distribution for Gaussian and lognormal x.

TABLE 7.9 Means and Variances from Example 7.10

	Distribution of x			
	Gaussian		Lognormal	
$z = x^2$	x	z	x	z
Mean	1	≈ 1.9	1	2
Variance	1	≈ 5.6	1	60

Sampling Methods Sampling methods have already demonstrated enormous improvement over linearized approaches, but these newer methods themselves are still evolving. Issues that need to be addressed include the following:

1. *UKF parameter tuning*. The UTs, which are perhaps the best approaches at present, have some "twiddle factors" for tuning the transforms to the application. Time-tested methodologies for tuning these parameters have yet to be formalized.
2. *Nonlinear-specific approaches*. Julier [120] recognized that the general nonlinear filtering problem cannot be solved with means and covariances alone, and he performed parametric analysis of sampling methods to develop a family of UT methods. However, just a few common types of nonlinearities occur quite often in applications. Rather than develop general-purpose nonlinear methods, it may be more fruitful and useful in the short term to concentrate on the more common types of nonlinearities. The following are some specific types of nonlinearities commonly encountered:
   (a) *State variable products.* These occur commonly in adaptive filtering applications, where some of the unknowns are elements of model parameters. They also occur in system identification problems, where the underlying process may be quasilinear but the unknowns of primary interest are the model parameters. These generally include the following:
      i. The state transition Φ_k, which usually changes slowly relative to the state variables. In some applications, this may include parameters u_i of an equivalent nonlinear transformation $\phi_k(x, u)$.
      ii. The measurement sensitivity matrix H_k, which also changes slowly relative to the state variables. It may be equivalent in some applications to the parameters u_i of an equivalent nonlinear transformation $h_k(x, u)$.
   (b) *Trigonometric nonlinearities*. These commonly occur in tracking sensors using bearing and elevation angles and in satellite orbit estimation problems. In the latter case, the problem is best solved by using Keplerian parameters as state variables. In the former case, it is quite possible that the parameters of the sampling methods can be optimized for the specific problem (i.e., optimizing the sampling method, itself, for trigonometric nonlinearities).
   (c) *Polynomial nonlinearities*. There are some applications for which nonlinear applications in polynomial approximations cannot be truncated

after the linear terms but can be truncated after the quadratic or cubic terms. The general-purpose second-order Kalman filter has already been derived by Bass et al. [23], and the third-order model has been derived by Wiberg and Campbell [282]. However, both of these approaches are extremely computationally complex compared to sampling methods. It is of interest whether there may be dependable sampling methods optimized for these problems, and with more acceptable computational complexity.

(d) *Rational nonlinearities*. The next common low order functions after polynomials are rational functions, which are ratios of polynomials. Many commonly used nonlinear functions are computed using Padé approximations, which are rational functions. (The practical implementation of the matrix exponential is an example.) Given that the (nonlinear) Riccati equation itself was linearized exactly by a rational substitution (and in the early eighteenth century, no less), it is surprising that rational methods have received so little attention in nonlinear Kalman filtering.

Status of EKF Sampling methods have pushed the envelope of applicability of Kalman filtering to more highly nonlinear problems, and the UKF works as well as the EKF for quasilinear problems. This has led some to call EKF "overextended Kalman filtering" or simply "flawed" as an estimation method, despite the considerable success of EKF for nearly half a century.

When there are perfectly good quasilinear models for an application, the EKF is a reliable and accurate estimator. There are still many important nonlinear problems (e.g., GNSS navigation) for which EKF works very well, and—thanks to symbolic math programs—taking partial derivatives is not only easy but very reliable.

The problem is that one cannot apply the EKF to *any* nonlinear problem and guarantee success. Users must always verify that quasilinear approximation is justified before using it, as they must verify the accuracy of the UKF on any particular problem before accepting its results. All nonlinear estimators are not perfectly reliable on all nonlinear problems, and users must *always* apply other means (e.g., Monte Carlo analysis) to verify their accuracy and reliability before accepting their results.

PROBLEMS

7.1 A scalar stochastic sequence x_k is given by

$$\begin{aligned} x_k &= -0.1x_{k-1} + \cos x_{k-1} + w_{k-1}, \quad z_k = x_k^2 + v_k, \\ Ew_k &= 0 = Ev_k, \quad \text{cov}w_k = \Delta(k_2 - k_1), \quad \text{cov}v_k = 0.5\Delta(k_2 - k_1), \\ Ex_0 &= 0, \quad P_0 = 1, \quad x_k^{\text{nom}} = 1. \end{aligned}$$

Determine the linearized and EKF estimator equations.

7.2 A scalar stochastic process $x(t)$ is given by

$$\begin{aligned}
\dot{x}(t) &= -0.5x^2(t) + w(t),\\
z(t) &= x^3(t) + v(t),\\
Ew(t) &= Ev(t) = 0\\
\text{cov}w_t &= \delta(t_1 - t_2), \quad \text{cov}v(t) = 0.5\delta(t_1 - t_2),\\
Ex_0 &= 0, \quad P_0 = 1, \quad x^{\text{nom}} = 1.
\end{aligned}$$

Determine the linearized and EKF estimator equations.

7.3 **(a)** Verify the results of Example 7.3 (noise-free simulation data).

(b) Estimate the states from a noisy data.

(c) Compare the results of linearized Kalman filters and EKF.

Assume that the plant noise is normally distributed with mean zero and covariance 0.2 and the measurement noise is normally distributed with mean zero and covariance 0.001.

7.4 Derive the linearized and EKF equations for the following equations:

$$x_k = f(x_{k-1}, k-1) + Gw_{k-1}, \quad z_k = h(x_k, k) + v_k.$$

7.5 Given the following plant and measurement model for a scalar dynamic system:

$$\begin{aligned}
\dot{x}(t) &= ax(t) + w(t), \quad z(t) = x(t) + v(t),\\
w(t) &\sim \mathcal{N}(0,1), \quad v(t) \sim \mathcal{N}(0,2)\\
Ex(0) &= 1\\
Ew(t)v(t) &= 0\\
P(0) &= 2,
\end{aligned}$$

assume an unknown constant parameter a and derive an estimator for a, given $z(t)$.

7.6 Let $\mathbf{r}$ represent the position vector to a magnet with dipole moment vector $\mathbf{m}$. The magnetic field vector $\mathbf{B}$ at the origin of the coordinate system in which $\mathbf{r}$ is measured is given by the formula

$$\mathbf{B} = \frac{\mu_0}{4\pi|\mathbf{r}|^5}\{3\mathbf{r}\mathbf{r}^{\mathrm{T}} - |\mathbf{r}|^2\mathbf{I}\}\mathbf{m} \tag{7.182}$$

in SI units.

(a) Derive the measurement sensitivity matrix for $\mathbf{Z} = \mathbf{B(r)}$ as the measurement and $\mathbf{m}$ as the state vector.

(b) Derive the sensitivity matrix for $\mathbf{r}$ as the state vector.

(c) If $\mathbf{r}$ is known but $\mathbf{m}$ is to be estimated from measurements of $\mathbf{B}$, is the estimation problem linear?

7.7 Generate the error covariance results for the plant and measurement models given in Example 7.3 with the appropriate values of process and measurement noise covariance and initial state estimation error covariance.

7.8 Generate the error covariance results for the plant and measurement models given in Example 7.4 with the appropriate values of process and measurement noise covariance and initial state estimation error covariance.

7.9 This problem is taken from reference [252]. The equations of motion for the space vehicle are

$$\ddot{r} - r\dot{\theta}^2 + \frac{k}{r^2} = w_r(t), \quad r\ddot{\theta} + 2\dot{r}\dot{\theta} = w_\theta(t),$$

where r is range, θ is bearing angle, k is a constant, and $w_r(t)$ and $w_\theta(t)$ are small random forcing functions in the r and θ directions. The observation equation is given by

$$z(t) = \begin{bmatrix} \sin^{-1}\left(\frac{R_e}{r}\right) \\ \alpha_0 - \theta \end{bmatrix}$$

where $R_e =$ earth radius and α_0 is a constant.
Linearize these equations about $r_{\text{nom}} = R_0$ and $\theta_{\text{nom}} = w_0 t$.

7.10 Let the 3×3 covariance matrix

$$P = \begin{bmatrix} 11 & 7 & -9 \\ 7 & 11 & -9 \\ -9 & -9 & 27 \end{bmatrix}. \tag{7.183}$$

(a) Use the built-in MATLAB function `chol` to compute the lower-triangular Cholesky factor of P. Note that the MATLAB command `C = chol(P)` returns an upper triangular matrix C such that $C^T C = P$, in which case C^T is a lower triangular Cholesky factor.

(b) Use the MATLAB function `utchol` on the CD to compute the upper triangular Choleksy factor of P.

(c) Use the MATLAB function `svd` to compute its symmetric square root, which is also a Cholesky factor. (Note that `[U,Lambda,V] = svd(P)` returns V such that $P = U\Lambda V^T$, where $V = U$.) The square root

of P is then $S_P = U\Lambda^{(1/2)}U^T$, where $S_P = C$ is also a symmetric Cholesky factor of P. Multiply your result by itself to verify your answer.

(d) What are the maximum and minimum eigenvalues of P?

(e) What is the condition number of P?

(The value of P in Equation 7.183 was used to generate the equiprobability ellipsoid shown in Figure 7.1.)

7.11 For each of the Cholesky factors in the previous problem, compute the corresponding $2n + 1 = 7$ sigma points for the UT with $\kappa = 0$. Assume that $\hat{x} = 0$, and do not forget to multiply the Cholesky factors by $\sqrt{n} = \sqrt{3}$ to obtain the properly scaled sigma points.

7.12 Using the three sets of sigma points from the previous problem, write a MATLAB program to compute the resulting means and variances obtained by transforming the sigma points through the nonlinear function

$$\vec{f}\left(\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}\right) = \begin{bmatrix} x_2 x_3 \\ x_3 x_1 \\ x_1 x_2 \end{bmatrix}.$$

7.13 Compare the resulting means from the previous problem to the exact mean

$$\mathrm{E}\left(\begin{bmatrix} x_2 x_3 \\ x_3 x_1 \\ x_1 x_2 \end{bmatrix}\right) = \begin{bmatrix} p_{2,3}(1 - p_{2,3}^2/p_{2,2}/p_{3,3})^{-3/2} \\ p_{3,1}(1 - p_{3,1}^2/p_{3,3}/p_{1,1})^{-3/2} \\ p_{1,2}(1 - p_{1,2}^2/p_{1,1}/p_{2,2})^{-3/2} \end{bmatrix}$$

for Gaussian initial distributions with zero mean and covariance P. Use the value of P from Equation 7.183 in this calculation.

7.14 Rework Example 7.5 using the symmetric UKF implementation with $2n$ samples. Use `utchol` to compute upper triangular Cholesky factors and the MATLAB function `ode45` (fourth- to fifth-order Runge–Kutta integration) to propagate the samples forward in time. Use the EKF approximation to compute Q_k from $\dot{Q} = FQF^T + Q_t$ with $Q(t_{k-1}) = 0$ and $F = \dfrac{\partial \dot{x}}{\partial x}$.

8

PRACTICAL CONSIDERATIONS

"The time has come," the Walrus said,

"To talk of many things:

Of shoes—and ships—and sealing wax—

Of cabbages—and kings—

And why the sea is boiling hot—

And whether pigs have wings."

From "The Walrus and the Carpenter" in *Through the Looking Glass*, 1872
—Lewis Carroll [Charles Lutwidge Dodgson] (1832–1898)

8.1 CHAPTER FOCUS

The discussion turns now to what might be called Kalman filter *engineering*, the body of applicable knowledge that has evolved through practical experience in the use and misuse of the Kalman filter. The material in the previous two chapters (square-root and nonlinear filtering) has also evolved in this way and is part of the same general subject. Here, however, the discussion includes many more matters of practice than nonlinearities and finite-precision arithmetic.

Kalman Filtering: Theory and Practice using MATLAB®, Third Edition. By Mohinder S. Grewal and Angus P. Andrews

8.1.1 Main Points to Be Covered

1. *Roundoff errors are not the only causes for the failure of the Kalman filter to achieve its theoretical performance*. There are diagnostic methods for identifying causes and remedies for other common patterns of misbehavior.
2. *Prefiltering to reduce computational requirements*. If the dynamics of the measured variables are "slow" relative to the sampling rate, then a simple prefilter can reduce the overall computational requirements without sacrificing performance.
3. *Detection and rejection of anomalous sensor data*. The inverse of the matrix $(HPH^{\mathrm{T}} + R)$ characterizes the probability distribution of the innovation $z - H\hat{x}$ and may be used to test for exogenous measurement errors, such as those resulting from sensor or transmission malfunctions.
4. *Statistical design of sensor and estimation systems*. The covariance equations of the Kalman filter provide an analytical basis for the predictive design of systems to estimate the state of dynamic systems. They may also be used to obtain suboptimal (but feasible) observation scheduling.
5. *Testing for asymptotic stability*. The relative robustness of the Kalman filter against minor modeling errors is due, in part, to the asymptotic stability of the Riccati equations defining performance.
6. *Model simplifications to reduce computational requirements*. A dual-state filter implementation can be used to analyze the expected performance of simplified Kalman filters based on simplifying the dynamic system model and/or the measurement model. These analyses characterize the trade-offs between performance and computational requirements.
7. *Memory and throughput requirements*. These computational requirements are represented as functions of "problem parameters" such as the dimensions of state and measurement vectors.
8. *Off-line processing to reduce on-line computational requirements*. Except in extended (nonlinear) Kalman filtering, the gain computations do not depend upon the real-time data. Therefore, they can be precomputed to reduce the real-time computational load.
9. *Innovations analysis* is a rather simple check for symptoms of mismodeling.

8.2 DETECTING AND CORRECTING ANOMALOUS BEHAVIOR

8.2.1 Convergence, Divergence, and "Failure to Converge"

Definitions of Convergence and Divergence A sequence $\{\eta_k | k = 1, 2, 3, \ldots\}$ of real vectors η_k is said to *converge* to a *limit* η_∞ if, for every $\varepsilon > 0$, for some n, for all $k > n$, the norm of the differences $\|\eta_k - \eta_\infty\| < \varepsilon$. Let us use the expressions

$$\lim_{k \to \infty} \eta_k = \eta_\infty \text{ or } \eta_k \to \eta_\infty$$

to represent convergence. One vector sequence is said to converge to another vector sequence if their differences converge to the zero vector, and a sequence is said to converge[1] if, for every $\varepsilon > 0$, for some integer n, for all $k, \ell > n$, $\|\eta_k - \eta_\ell\| < \varepsilon$. *Divergence* is defined as convergence to ∞: for every $\varepsilon > 0$, for some integer n, for all $k > n$, $|\eta_k| > \varepsilon$. In that case, $||\eta_k||$ is said to *grow without bound*.

Nonconvergence This is a more common issue in the performance of Kalman filters than strict divergence. That is, the filter fails because it does not converge to the desired limit, although it does not necessarily diverge.

Dynamic and Stochastic Variables Subject to Convergence or Divergence The operation of a Kalman filter involves the following sequences that may or may not converge or diverge:

x_k, the sequence of actual state values;
$E\langle x_k x_k^{\mathrm{T}}\rangle$, the mean-squared state;
$\hat{x}_k$, the estimated state;
$\tilde{x}_k(-) = \hat{x}_k(-) - x_k$, the *a priori* estimation error;
$\tilde{x}_k(+) = \hat{x}_k(+) - x_k$, the *a posteriori* estimation error;
$P_k(-)$, the covariance of *a priori* estimation errors;
$P_k(+)$, the covariance of *a posteriori* estimation errors.

One may also be interested in whether or not the sequences $\{P_k(-)\}$ and $\{P_k(+)\}$ computed from the Riccati equations converge to the corresponding *true* covariances of the estimation error.

8.2.2 Use of the Riccati Equation to Predict Behavior

The covariance matrix of estimation uncertainty characterizes the theoretical performance of the Kalman filter. It is computed as an ancillary variable in the Kalman filter as the solution of a matrix Riccati equation with the given initial conditions. It is also useful for predicting performance. If its characteristic values are growing without bound, then the theoretical performance of the Kalman filter is said to be diverging. This can happen if the system state is unstable and unobservable, for example. This type of divergence is detectable by solving the Riccati equation to compute the covariance matrix.

The Riccati equation is not always well conditioned for numerical solution, and one may need to use the more numerically stable methods of Chapter 6 to obtain reasonable results. One can, for example, use eigenvalue–eigenvector decomposition of solutions to test their characteristic roots (they should be positive) and condition numbers. Condition numbers within one or two orders of magnitude of ε^{-1} (the reciprocal of the unit roundoff error in computer precision) are considered probable cause for concern and reason to use square-root methods.

[1]Such sequences are called *Cauchy sequences*, after Augustin Louis Cauchy (1789–1857).

8.2.3 Testing for Unpredictable Behavior

Not all filter divergence is predictable from the Riccati equation solution. Sometimes the actual performance does not agree with the theoretical performance.

One cannot measure estimation error directly except in simulations, so one must find other means to check on estimation accuracy. Whenever the estimation error is deemed to differ significantly from its expected value (as computed by the Riccati equation), the filter is said to diverge from its predicted performance. We will now consider how one might go about detecting divergence.

Examples of typical behaviors of Kalman filters are shown in Figure 8.1, which is a multiplot of the estimation errors on 10 different simulations of a filter implementation with independent pseudorandom-error sequences. Note that each time the filter is run, different estimation errors $\tilde{x}(t)$ result, even with the same initial condition $\hat{x}(0)$. Also note that at any particular time, the average estimation error (across the ensemble of simulations) is approximately zero,

$$\frac{1}{N}\sum_{i=1}^{N}[\hat{x}_i(t_k) - x(t_k)] \approx \underset{i}{E}\langle\hat{x}_i(t_k) - x(t_k)\rangle = 0, \tag{8.1}$$

where N is the number of simulation runs and $\hat{x}_i(t_k) - x(t_k)$ is the estimation error at time t_k on the ith simulation run.

Monte Carlo analysis of Kalman filter performance uses many such runs to test that the ensemble mean estimation error is *unbiased* (i.e., has effectively zero mean) and that its ensemble covariance is in close agreement with the theoretical value computed as a solution of the Riccati equation.

Convergence of Suboptimal Filters In the suboptimal filters discussed in Section 8.5, the estimates can be biased. Therefore, in the analysis of suboptimal

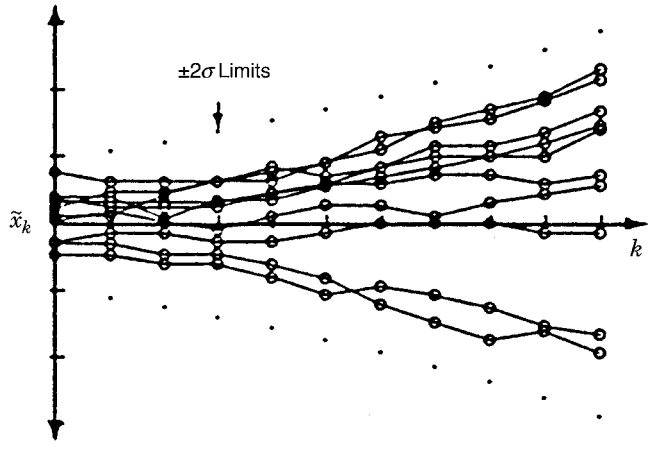


Figure 8.1 Dispersion of multiple runs.

filters, the behavior of $P(t)$ is not sufficient to define convergence. A suboptimal filter is said to converge if the covariance matrices converge,

$$\lim_{t\to\infty}[\text{trace}(P_{\text{sub-opt}} - P_{\text{opt}})] = 0, \tag{8.2}$$

and the asymptotic estimation error is unbiased,

$$\lim_{t\to\infty} \mathrm{E}[\tilde{x}(t)] = 0. \tag{8.3}$$

Example 8.1 (Behaviors of Continuous-Time Systems) Some typical behavior patterns of suboptimal filter convergence are depicted by the plots of $P(t)$ in Figure 8.2(a), and characteristics of systems with these symptoms are given here as examples.

Case A: Let a scalar continuous system equation be given by

$$\dot{x}(t) = Fx(t), \qquad F > 0, \tag{8.4}$$

in which the system is unstable, or

$$\dot{x}(t) = Fx(t) + w(t), \tag{8.5}$$

in which the system has driving noise and is unstable.

Case B: The system has constant steady-state uncertainty:

$$\lim_{t\to\infty} \dot{P}(t) = 0. \tag{8.6}$$

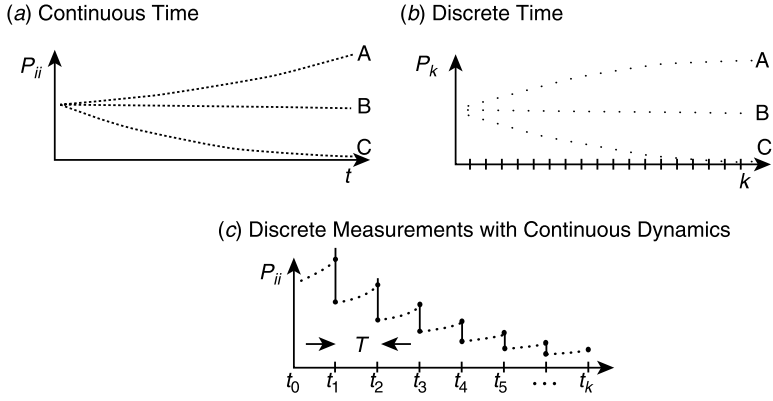


Figure 8.2 Asymptotic behaviors of estimation uncertainties.

Case C: The system is stable and has no driving noise:

$$\dot{x}(t) = -Fx(t), \qquad F > 0. \tag{8.7}$$

Example 8.2 (Behaviors of Discrete-Time Systems) Plots of P_k are shown in Figure 8.2(*b*) for the following system characteristics:

Case A: Effects of system driving noise and measurement noise are large relative to $P_0(t)$ (initial uncertainty).

Case B: $P_0 = P_\infty$ (Wiener filter).

Case C: Effects of system driving noise and measurement noise are small relative to $P_0(t)$.

Example 8.3 (Continuous System with Discrete Measurements) A scalar example of a behavior pattern of the covariance propagation equation ($P_k(-)$, $\dot{P}(t)$) and the covariance update equation $P_k(+)$,

$$\begin{aligned} \dot{x}(t) &= Fx(t) + w(t), \quad F < 0, \\ z(t) &= x(t) + v(t), \end{aligned}$$

is shown in Figure 8.2(*c*).

The following features may be observed in the behavior of $P(t)$:

1. Processing the measurement tends to reduce P.
2. Process noise covariance (Q) tends to increase P.
3. Damping in a stable system tends to reduce P.
4. Unstable system dynamics ($F > 0$) tend to increase P.
5. With white Gaussian measurement noise, the time between samples (T) can be reduced to decrease P.

The behavior of P represents a composite of all these effects (1–5), as shown in Figure 8.2(*c*).

Causes of Predicted Nonconvergence Nonconvergence of P predicted by the Riccati equation can be caused by

1. "natural behavior" of the dynamic equations or
2. nonobservability with the given measurements.

The following examples illustrate these behavioral patterns.

Example 8.4 (Natural Behavior) The natural behavior for P in some cases is for

$$\lim_{t\to\infty} P(t) = P_\infty \text{ (a constant).} \tag{8.8}$$

For example,

$$\left.\begin{array}{ll} \dot{x} = w, & \operatorname{cov}(w) = Q \\ z = x + v, & \operatorname{cov}(v) = R \end{array}\right) \Longrightarrow \begin{array}{ll} F = 0 & \text{in} \quad \dot{x} = Fx + Gw, \\ G = H = 1 & \quad\;\; z = Hx + v. \end{array} \tag{8.9}$$

Applying the continuous Kalman filter equations from Chapter 4, then,

$$\dot{P} = FP + PF^{\mathrm{T}} + GQG^{\mathrm{T}} - \overline{K}R\overline{K}^{\mathrm{T}}$$

and

$$\overline{K} = PH^{\mathrm{T}}R^{-1}$$

become

$$\dot{P} = Q - \overline{K}^2 R$$

and

$$\overline{K} = \frac{P}{R}$$

or

$$\dot{P} = Q - \frac{P^2}{R}.$$

The solution is

$$P(t) = \alpha\left(\frac{P_0\cosh(\beta t) + \alpha\sinh(\beta t)}{P_0\sinh(\beta t) + \alpha\cosh(\beta t)}\right), \tag{8.10}$$

where

$$\alpha = \sqrt{RQ}, \quad \beta = \sqrt{Q/R}. \tag{8.11}$$

Note that the solution of the Riccati equation converges to a finite limit:

1. $\lim_{t\to\infty} P(t) = \alpha > 0$, a finite but nonzero limit. (See Figure 8.3(*a*).)
2. This is no cause for alarm, and there is no need to remedy the situation if the asymptotic mean-squared uncertainty is tolerable. If it is *not* tolerable, then the remedy must be found in the hardware (e.g., by attention to the physical sources of R or Q—or both), not in software.

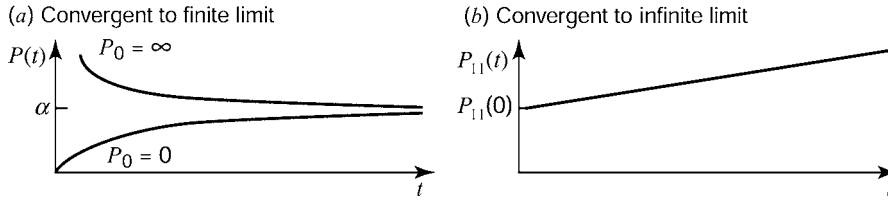


Figure 8.3 Behavior patterns of P.

Example 8.5 ("Structural" Unobservability) The filter is said to diverge at infinity if its limit is unbounded:

$$\lim_{t\to\infty} P(t) = \infty. \tag{8.12}$$

As an example in which this occurs, consider the system

$$\begin{aligned} \dot{x}_1 &= w, \\ \dot{x}_2 &= 0, \\ z &= x_2 + v, \end{aligned} \qquad \begin{aligned} &\operatorname{cov}(w) = Q, \\ &\operatorname{cov}(v) = R, \end{aligned} \tag{8.13}$$

with initial conditions

$$P_0 = \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix} = \begin{bmatrix} P_{11}(0) & 0 \\ 0 & P_{22}(0) \end{bmatrix}. \tag{8.14}$$

The continuous Kalman filter equations

$$\begin{aligned} \dot{P} &= FP + PF^{\mathrm{T}} + Q - \overline{K}R\overline{K}^{\mathrm{T}}, \\ \overline{K} &= PH^{\mathrm{T}}R^{-1} \end{aligned}$$

can be combined to give

$$\dot{P} = FP + PF^{\mathrm{T}} + Q - PH^{\mathrm{T}}R^{-1}HP, \tag{8.15}$$

or

$$\dot{p}_{11} = Q - \frac{p_{12}^2}{R}, \quad \dot{p}_{12} = -\frac{p_{12}p_{22}}{R}, \quad \dot{p}_{22} = -\frac{p_{22}^2}{R}, \tag{8.16}$$

the solution to which is

$$p_{11}(t) = p_{11}(0) + Qt, \quad p_{12}(t) = 0, \quad p_{22}(t) = \frac{p_{22}(0)}{1 + [p_{22}(0)/R]t}, \tag{8.17}$$

as plotted in Figure 8.3(b). The only remedy in this example is to alter or add measurements (sensors) to achieve observability.

Example 8.6 (Nonconvergence Due to "Structural" Unobservability) Parameter estimation problems have no state dynamics and no process noise. One might reasonably expect the estimation uncertainty to approach zero asymptotically as more and more measurements are made. However, it can still happen that the filter will not converge to absolute certainty. That is, the asymptotic limit of the estimation uncertainty

$$0 < \lim_{k \to \infty} P_k < \infty \tag{8.18}$$

is actually bounded away from zero uncertainty.

Parameter Estimation Model for Continuous Time Consider the two-dimensional parameter estimation problem

$$\left.\begin{aligned} &\dot{x}_1 = 0, \qquad \dot{x}_2 = 0, \quad P_0 = \begin{bmatrix} \sigma_1^2(0) & 0 \\ 0 & \sigma_2^2(0) \end{bmatrix}, \quad H = [\,1 \quad 1\,], \\ &z = H \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + v, \qquad \operatorname{cov}(v) = R, \end{aligned}\right\} \tag{8.19}$$

in which only the *sum* of the two state variables is measurable. The difference of the two state variables will then be unobservable.

Problem in Discrete Time This example also illustrates a difficulty with a standard shorthand notation for discrete-time dynamic systems: the practice of using subscripts to indicate discrete time. Subscripts are more commonly used to indicate components of vectors. The solution here is to move the component indices "upstairs" and make them superscripts. (This approach works here only because the problem is linear. Therefore, one does not need superscripts to indicate powers of the components.) For these purposes, let x_k^i denote the ith component of the state vector at time t_k. The continuous form of the parameter estimation problem can then be "discretized" to a model for a discrete Kalman filter (for which the state transition matrix is the identity matrix; see Section 4.2):

$$x_k^1 = x_{k-1}^1 \; (x^1 \text{ is constant}), \tag{8.20}$$

$$x_k^2 = x_{k-1}^2 \; (x^2 \text{ is constant}), \tag{8.21}$$

$$z_k = \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} x_k^1 \\ x_k^2 \end{bmatrix} + v_k. \tag{8.22}$$

Let

$$\hat{x}_0 = 0.$$

The estimator then has two sources of information from which to form an optimal estimate of x_k:

1. the *a priori* information in $\hat{x}_0$ and P_0 and
2. the measurement sequence $z_k = x_k^1 + x_k^2 + v_k$ for $k = 1, 2, 3, \ldots$.

In this case, the best the optimal filter can do with the measurements is to "average out" the effects of the noise sequence $v_1, \ldots, v_k$. One might expect that an infinite number of measurements (z_k) would be equivalent to *one* noise-free measurement, that is,

$$z_1 = (x_1^1 + x_1^2), \quad \text{where } v_1 \to 0 \quad \text{and} \quad R = \text{cov}(v_1) \to 0. \tag{8.23}$$

Estimation Uncertainty from a Single Noise-Free Measurement By using the discrete filter equations with one stage of estimation on the measurement z_1, one can obtain the gain in the form

$$\overline{K}_1 = \begin{bmatrix} \dfrac{\sigma_1^2(0)}{(\sigma_1^2(0) + \sigma_2^2(0) + R)} \\ \dfrac{\sigma_2^2(0)}{\sigma_1^2(0) + \sigma_2^2(0) + R} \end{bmatrix}. \tag{8.24}$$

The estimation uncertainty covariance matrix can then be shown to be

$$P_1(+) = \begin{bmatrix} \dfrac{\sigma_1^2(0)\sigma_2^2(0) + R\sigma_1^2(0)}{\sigma_1^2(0) + \sigma_2^2(0) + R} & \dfrac{-\sigma_1^2(0)\sigma_2^2(0)}{\sigma_1^2(0) + \sigma_2^2(0) + R} \\ \dfrac{-\sigma_1^2(0)\sigma_2^2(0)}{\sigma_1^2(0) + \sigma_2^2(0) + R} & \dfrac{\sigma_1^2(0)\sigma_2^2(0) + R\sigma_2^2(0)}{\sigma_1^2(0) + \sigma_2^2(0) + R} \end{bmatrix} \equiv \begin{bmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{bmatrix}, \tag{8.25}$$

where the *correlation coefficient* (defined in Equation 3.157) is

$$\rho_{12} = \frac{p_{12}}{\sqrt{p_{11}p_{22}}} = \frac{-\sigma_1^2(0)\sigma_2^2(0)}{\sqrt{[\sigma_1^2(0)\sigma_2^2(0) + R\sigma_1^2(0)][\sigma_1^2(0)\sigma_2^2(0) + R\sigma_2^2(0)]}}, \tag{8.26}$$

and the state estimate is

$$\hat{x}_1 = \hat{x}_1(0) + \overline{K}_1[z_1 - H\hat{x}_1(0)] = [I - \overline{K}_1 H]\, \hat{x}_1(0) + \overline{K}_1 z_1. \tag{8.27}$$

However, for the *noise-free* case,

$$v_1 = 0, \quad R = \text{cov}(v_1) = 0,$$

the correlation coefficient is

$$\rho_{12} = -1, \tag{8.28}$$

and the estimates for $\hat{x}_1(0) = 0$,

$$\hat{x}_1^1 = \left(\frac{\sigma_1^2(0)}{\sigma_1^2(0) + \sigma_2^2(0)}\right)(x_1^1 + x_1^2),$$
$$\hat{x}_1^2 = \left(\frac{\sigma_2^2(0)}{\sigma_1^2(0) + \sigma_2^2(0)}\right)(x_1^1 + x_1^2),$$

are totally insensitive to the difference $x_1^1 - x_1^2$. As a consequence, the filter will *almost never get the right answer*! This is a fundamental characteristic of the problem, however, and is not attributable to the design of the filter. There are *two* unknowns (x_1^1 and x_1^2) and *one* constraint:

$$z_1 = (x_1^1 + x_1^2). \tag{8.29}$$

Conditions for Serendipitous Design The conditions under which the filter will still get the right answer can easily be derived. Because x_1^1 and x_1^2 are constants, their ratio constant

$$C \stackrel{\text{def}}{=} \frac{x_1^2}{x_1^1} \tag{8.30}$$

will also be a constant, such that the sum

$$\begin{aligned} x_1^1 + x_1^2 &= (1 + C)x_1^1 \\ &= \left(\frac{1 + C}{C}\right)x_1^2. \end{aligned}$$

Then

$$\hat{x}_1^1 = \left(\frac{\sigma_1^2(0)}{\sigma_1^2(0) + \sigma_2^2(0)}\right)[(1 + C)x_1^1] = x_1^1 \quad \text{only if} \quad \frac{\sigma_1^2(0)(1 + C)}{\sigma_1^2(0) + \sigma_2^2(0)} = 1,$$
$$\hat{x}_1^2 = \left(\frac{\sigma_2^2(0)}{\sigma_1^2(0) + \sigma_2^2(0)}\right)\left(\frac{1 + C}{C}\right)x_1^2 = x_1^2 \quad \text{only if} \quad \frac{\sigma_2^2(0)(1 + C)}{[\sigma_1^2(0) + \sigma_2^2(0)](C)} = 1.$$

Both of these conditions are satisfied *only if*

$$\frac{\sigma_2^2(0)}{\sigma_1^2(0)} = C = \frac{x_1^2}{x_1^1} \geq 0, \tag{8.31}$$

because $\sigma_1^2(0)$ and $\sigma_2^2(0)$ are nonnegative numbers.

Likelihood of Serendipitous Design For the filter to obtain the right answer, it would be necessary that

1. x_1^1 and x_1^2 have the *same sign* and
2. it is known that their ratio $C = x_1^2/x_1^1$.

Since both of these conditions are rarely satisfied, the filter estimates would rarely be correct.

What Can Be Done About It The following methods can be used to detect nonconvergence due to this type of structural unobservability:

- Test the system for observability using the observability theorems in Section 2.5.
- Look for perfect correlation coefficients ($\rho = \pm 1$) and be very suspicious of high correlation coefficients (e.g., $|\rho| > 0.9$).
- Perform eigenvalue–eigenvector decomposition of P to test for negative characteristic values or a large condition number. (This is a better test than correlation coefficients to detect unobservability.)
- Test the filter on a system simulator with *noise-free outputs* (measurements).

Remedies for this problem include

- attaining observability by adding another type of measurement or
- defining $\acute{x} \equiv x^1 + x^2$ as the only state variable to be estimated.

Example 8.7 (Unobservability Caused by Poor Choice of Sampling Rate) The problem in Example 7.6 might be solved by using an additional measurement—or by using a measurement with a time-varying sensitivity matrix. Next, consider what can go wrong even with time-varying measurement sensitivities if the sampling rate is chosen badly. For that purpose, consider the problem of estimating unknown (constant states) with both constant and sinusoidal measurement sensitivity matrix components:

$$H(t) = [1 \quad \cos(\omega t)],$$

as plotted in Figure 8.4. The equivalent model for use in the discrete Kalman filter is

$$x_k^1 = x_{k-1}^1, \quad x_k^2 = x_{k-1}^2, \quad H_k = H(kT), \quad z_k = H_k \begin{bmatrix} x_k^1 \\ x_k^2 \end{bmatrix} + v_k,$$

where T is the intersample interval.

What Happens When Murphy's Law Takes Effect With the choice of intersampling interval as $T = 2\pi/\omega$ and $t_k = kT$, the components of the measurement

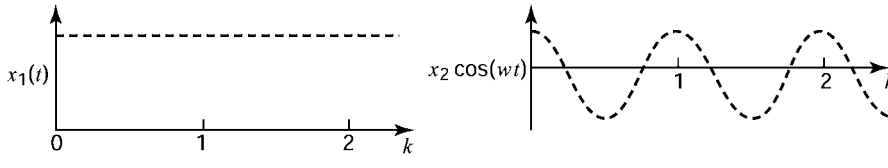


Figure 8.4 Aliased measurement components.

sensitivity matrix become equal and constant:

$$\begin{aligned} H_k &= [1 \;\; \cos(\omega k T)] \\ &= [1 \;\; \cos(2\pi k)] \\ &= [1 \;\; 1], \end{aligned}$$

as shown in Figure 8.4. (This is the way many engineers discover aliasing.) The states x^1 and x^2 are *unobservable* with this choice of sampling interval (see Figure 8.4). With this choice of sampling times, the system and filter behave as in the previous example.

Methods for detecting and correcting unobservability include those given in Example 7.6 plus the more obvious remedy of changing the sampling interval T to obtain observability; for example,

$$T = \frac{\pi}{\omega} \tag{8.32}$$

is a better choice.

Causes of Unpredicted Nonconvergence Unpredicted nonconvergence may be caused by

1. bad data,
2. numerical problems, or
3. mismodeling.

Example 8.8 (Unpredicted Nonconvergence Due to Bad Data) Bad data are caused by something going wrong, which is almost sure to happen in real-world applications of Kalman filtering. These verifications of Murphy's law occur principally in two forms:

- The initial estimate is badly chosen, for example,

$$|\hat{x}(0) - x|^2 = |\tilde{x}|^2 \gg \text{trace} P_0. \tag{8.33}$$

- The measurement has an exogenous component (a mistake, not an error) that is excessively large, for example,

$$|v|^2 \gg \text{trace} R. \tag{8.34}$$

Asymptotic Recovery from Bad Data In either case, if the system is truly linear, the Kalman filter will (in theory) recover in finite time as it continues to use measurements z_k to estimate the state x. (The best way is to prevent bad data from getting into the filter in the first place!) See Figure 8.5.

Practical Limitations of Recovery Often, in practice, the recovery is not adequate in finite time. The interval $(0, T)$ of measurement availability is fixed and may be too

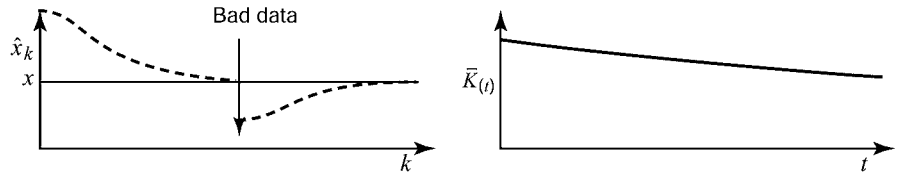


Figure 8.5 Asymptotic recovery from bad data.

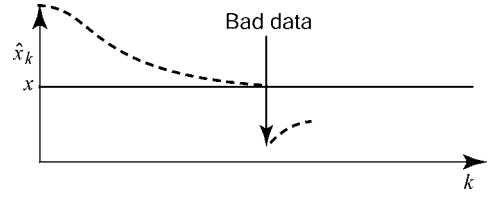


Figure 8.6 Failure to recover in a short period.

short to allow sufficient recovery (see Figure 8.6). The normal behavior of the gain matrix $\overline{K}$ may be converging too rapidly toward its steady-state value of $\overline{K} = 0$. (See Figure 8.7.)

Remedies for Heading Off Bad Data

- Inspection of $P(t)$ and $\overline{K}(t)$ is useless, because they are not affected by data.
- Inspection of the state estimates $\hat{x}(t)$ for sudden jumps (*after* a bad measurement has already been used by the filter) is sometimes used, but it still leaves the problem of undoing the damage to the estimate after it has been done.
- Inspection of the "innovations" vector $[z - H\hat{x}]$ for sudden jumps or large entries (*before* bad measurement is processed by the filter) is much more useful, because the discrepancies can be interpreted probabilistically, and the data can be discarded before they have spoiled the estimate (see Section 8.3).

The best remedy for this problem is to implement a *bad-data detector* to reject the bad data before they contaminate the estimate. If this is to be done in real time, it is sometimes useful to save the bad data for off-line examination by an *exception handler* (often a human, but sometimes a second-level data analysis program) to locate and remedy the causes of the bad data that are occurring.

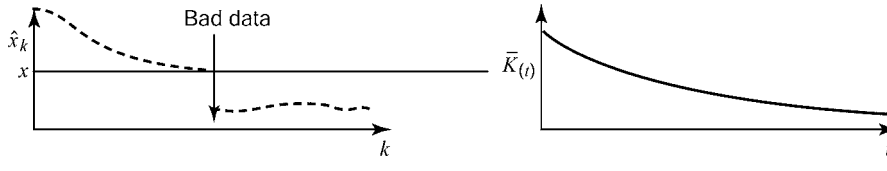


Figure 8.7 Failure to recover due to gain decay.

Artificially Increasing Process Noise Covariance to Improve Bad Data Recovery If bad data are detected after they have been used, one can keep the filter "alive" (to pay more attention to subsequent data) by increasing the process noise covariance Q in the system model assumed by the filter. Ideally, the new process noise covariance should reflect the actual measurement error covariance, including the bad data as well as other random noise.

Example 8.9 (Nonconvergence Due to Numerical Problems) This is sometimes detected by observing impossible P_k behavior. The terms on the main diagonal of P_k may become negative, or larger, immediately after a measurement is processed rather than immediately before, that is, $\sigma(+) > \sigma(-)$. A less obvious (but detectable) failure mode is for the characteristic values of P to become negative. This can be detected by eigenvalue–eigenvector decomposition of P. Other means of detection include using simulation (with known states) to compare estimation errors with their estimated covariances. One can also use double precision in place of single precision to detect differences due to precision. Causes of numerical problems can sometimes be traced to inadequate wordlength (precision) of the host computer. These problems tend to become worse with larger numbers of state variables.

Remedies for Numerical Problems These problems have been treated by many "brute-force" methods, such as using higher precision (e.g., double instead of single). One can try reducing the number of states by merging or eliminating unobservable states, eliminating states representing "small effects," or using other suboptimal filter techniques such as decomposition into lower dimensional state spaces.

Possible remedies include the use of *more numerically stable methods* (to obtain more computational accuracy with the same computer precision) and the use of *higher precision*. The latter approach (higher precision) will increase the execution time but will generally require less reprogramming effort. One can sometimes use a better algorithm to improve the accuracy of matrix inverse $(HPH^{\mathrm{T}} + R)^{-1}$ (e.g., the Cholesky decomposition method shown in Chapter 6) or eliminate the inverse altogether by diagonalizing R and processing the measurements sequentially (also shown in Chapter 6).

8.2.4 Effects Due to Mismodeling

Lying to the Filter The Kalman gain and the covariance matrix P are correct *if* the models used in computing them are correct. With mismodeling, the P matrix can be erroneous and of little use in detecting nonconvergence, or P can even converge to zero while the state estimation error $\tilde{x}$ is actually diverging. (It happens.)

The problems that result from bad initial estimates of x or P have already been addressed. There are four other types that will now be addressed:

1. unmodeled state variables of the dynamic system,
2. unmodeled process noise,

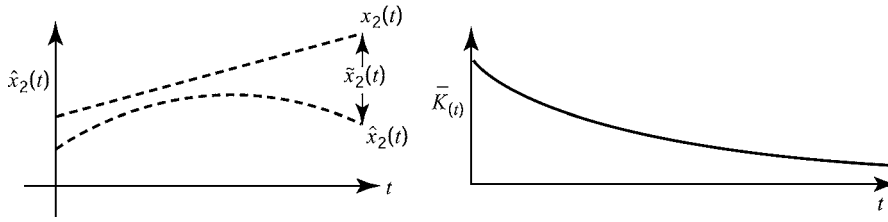


Figure 8.8 Divergence due to mismodeling.

3. errors in the dynamic coefficients or the state transition matrix, and
4. overlooked nonlinearities.

Example 8.10 (Nonconvergence Caused by Unmodeled State Variables) Consider the following example:

Real World (Creeping State)	Kalman Filter Model (Constant State)
$\dot{x}_1 = 0$	$\dot{x}_2 = 0$
$\dot{x}_2 = x_1$	$z = x_2 + v$
$z = x_2 + v$	
$\Rightarrow x_2(t) = x_2(0) + x_1(0)t$	$\Rightarrow x_2(t) = x_2(0)$

(8.35)

for which, in the filter model, the Kalman gain $\overline{K}(t) \to 0$ as $t \to \infty$. The filter is unable to provide feedback for the error in the estimate of $x_2(t)$ as it grows in time (even if it grows slowly). Eventually, $\tilde{x}_2(t) = \hat{x}_2(t) - x_2(t)$ diverges, as shown in Figure 8.8.

Detecting Unmodeled State Dynamics by Fourier Analysis of the Filter Innovations It is difficult to diagnose unmodeled state variables unless all other causes of nonconvergence have been ruled out. If there is high confidence in the model being used, then simulation can be used to rule out any of the other causes identified above. Once these other causes have been eliminated, Fourier analysis of the filter innovations (the prediction errors $\{z_k - H_k\hat{x}_k\}$) can be useful for spotting characteristic frequencies of the unmodeled state dynamics. If the filter is modeled properly, then the innovations should be uncorrelated, having a power spectral density (PSD) that is essentially flat. Persistent peaks in the PSD indicate the characteristic frequencies of unmodeled effects. These peaks can be at zero frequency, which indicates a bias error in the innovations.

Remedies for Unmodeled State Variables The best cure for nonconvergence caused by unmodeled states is to correct the model, but this not always easy to do. As an *ad hoc* fix, additional fictitious process noise can be added to the system model assumed by the Kalman filter.

Example 8.11 (Adding Fictitious Process Noise to the Kalman Filter Model) Continuing with the continuous-time problem of Example 8.10, consider the alternative Kalman filter model

$$\dot{x}_2(t) = w(t), \quad z(t) = x_2(t) + v(t).$$

"Type 1 Servo" Behavior The behavior of this filter can be analyzed by applying the continuous Kalman filter equations from Chapter 4, with parameters

$$F = 0, \quad H = 1, \quad G = 1,$$

transforming the general Riccati differential equation

$$\begin{aligned}\dot{P} &= FP + PF^{\mathrm{T}} - PH^{\mathrm{T}}R^{-1}HP + GQG^{\mathrm{T}} \\ &= \frac{-P^2}{R} + Q\end{aligned}$$

into a scalar equation with steady-state solution (to $\dot{P} = 0$)

$$P(\infty) = \sqrt{RQ}.$$

The steady-state Kalman gain

$$\overline{K}(\infty) = \frac{P(\infty)}{R} = \sqrt{\frac{Q}{R}}. \tag{8.36}$$

The equivalent steady-state model of the Kalman filter can now be formulated as follows:

$$\dot{\hat{x}}_2 = F\hat{x}_2 + \overline{K}[z - H\hat{x}_2]. \tag{8.37}$$

Here, $F = 0$, $H = 1$, $\overline{K} = \overline{K}(\infty)$, and $\hat{x} = \hat{x}_2$, so that

$$\dot{\hat{x}}_2 + \overline{K}(\infty)\hat{x}_2 = \overline{K}(\infty)z. \tag{8.38}$$

The steady-state response of this estimator can be determined analytically by taking Laplace transforms:

$$[s + \overline{K}(\infty)]\hat{x}_2(s) = \overline{K}(\infty)z(s) \tag{8.39}$$

$$\Rightarrow \frac{\hat{x}_2(s)}{z(s)} = \frac{\overline{K}(\infty)}{s + \overline{K}(\infty)}. \tag{8.40}$$

Figure 8.9(*a*) shows a type 1 servo. Its steady-state following error (even in the noise-free case) is not zero in the real-world case:

$$z(t) = x_2(t) = x_2(0) + x_1(0)t \quad \text{with} \quad v = 0.$$

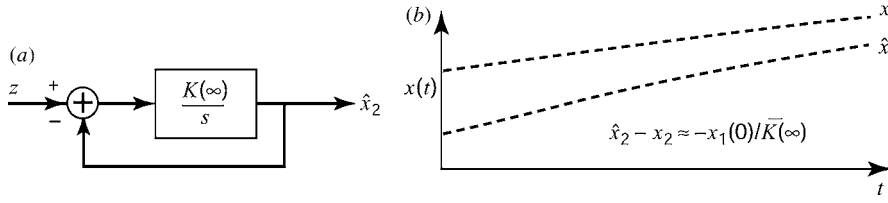


Figure 8.9 Type 1 servo (*a*) and estimates (*b*).

Taking the Laplace transform of the equation yields

$$z(s) = x_2(s) = \frac{x_2(0)}{s} + \frac{x_1(0)}{s^2}.$$

The error in $x_2(t)$ is

$$\tilde{x}_2(t) = \hat{x}_2(t) - x_2(t).$$

Taking the Laplace transform of the equation and substituting the value of $\hat{x}_2(s)$ from Equation 8.40 give

$$\begin{aligned}\tilde{x}_2(s) &= \frac{\overline{K}(\infty)}{s + \overline{K}(\infty)} x_2(s) - x_2(s) \\ &= \left[-\frac{s}{s + \overline{K}(\infty)} \right] x_2(s).\end{aligned}$$

Applying the final-value theorem, one gets

$$\begin{aligned}\tilde{x}_2(\infty) &= [\hat{x}_2(\infty) - x_2(\infty)] = \lim_{s \to 0} s[\hat{x}_2(s) - x_2(s)] \\ &= \lim_{s \to 0} s \left[-\frac{s}{s + \overline{K}(\infty)} \right] [x_2(s)] \\ &= \lim_{s \to 0} s \left[-\frac{s}{s + \overline{K}(\infty)} \right] \left[\frac{x_2(0)}{s} + \frac{x_1(0)}{s^2} \right] \\ &= -\frac{x_1(0)}{\overline{K}(\infty)} (\text{a bias}).\end{aligned}$$

This type of behavior is shown in Figure 8.9(*b*).

If the steady-state bias in the estimation error is unsatisfactory with the approach in Example 7.11, one can go one step further by adding another state variable and fictitious process noise to the system model assumed by the Kalman filter.

Example 8.12 (Effects of Adding States and Process Noise to the Kalman Filter Model) Suppose that the model of Example 7.11 was modified to the

following form:

Real World	Kalman Filter Model
$\dot{x}_1 = 0$	$\dot{x}_1 = w$
$\dot{x}_2 = x_1$	$\dot{x}_2 = x_1$
$z = x_2 + v$	$z = x_2 + v$

(8.41)

That is, $x_2(t)$ is now modeled as an *integrated random walk*. In this case, the steady-state Kalman filter has an additional integrator and behaves like a type 2 servo (see Figure 8.10(*a*)). The type 2 servo has zero steady-state following error to a ramp. However, its transient response may become more sluggish and its steady-state error due to noise is not zero, the way it would be with the real world correctly modeled. Here,

$$F = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}, \quad G = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad H = [0 \quad 1], \quad Q = \text{cov}(w), \quad R = \text{cov}(v) \tag{8.42}$$

$$\dot{P} = FP + PF^{\mathrm{T}} + GQG^{\mathrm{T}} - PH^{\mathrm{T}}R^{-1}HP \text{ and } \overline{K} = PH^{\mathrm{T}}R^{-1} \tag{8.43}$$

become in the steady state

$$\left.\begin{aligned} \dot{p}_{11} &= Q - \frac{p_{12}^2}{R} = 0 \\ \dot{p}_{12} &= p_{11} - \frac{p_{12}p_{22}}{R} = 0 \\ \dot{p}_{22} &= 2p_{12} - \frac{p_{22}^2}{R} = 0 \end{aligned}\right\} \Rightarrow \begin{aligned} p_{12}(\infty) &= \sqrt{RQ}, \\ p_{22}(\infty) &= \sqrt{2}(R^3Q)^{1/4}, \\ p_{11}(\infty) &= \sqrt{2}(Q^3R)^{1/4}, \\ \overline{K}(\infty) &= \begin{bmatrix} \sqrt{\dfrac{Q}{R}} \\ \sqrt{2}\sqrt[4]{\dfrac{Q}{R}} \end{bmatrix} = \begin{bmatrix} \overline{K}_1(\infty) \\ \overline{K}_2(\infty) \end{bmatrix}, \end{aligned} \tag{8.44}$$

and these can be Laplace transformed to yield

$$\hat{x}(s) = [sI - F + \overline{K}(\infty)H]^{-1}\overline{K}(\infty)z(s). \tag{8.45}$$

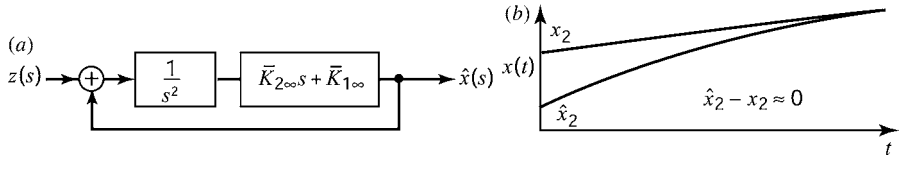


Figure 8.10 Type 2 servo (*a*) and servo estimates (*b*).

In component form, this becomes

$$\hat{x}_2(s) = \frac{(\overline{K}_2 s + \overline{K}_1)/s^2}{1 + (\overline{K}_2 s + \overline{K}_1)/s^2} z(s). \tag{8.46}$$

The resulting steady-state following error to a ramp (in the noise-free case) is easily determined:

$$\begin{aligned} z(t) &= x_2(t) = x_2(0) + x_1(0)t, \quad v = 0, \\ \tilde{x}_2(s) &= \hat{x}_2(s) - x_2(s) = -\left(\frac{s^2}{s^2 + \overline{K}_2 s + \overline{K}_1}\right) x_2(s), \\ \tilde{x}_2(\infty) &= \lim_{s \to 0} s \left(\frac{-s^2}{s^2 + \overline{K}_2 s + \overline{K}_1}\right)\left(\frac{x_2(0)}{s} + \frac{x_1(0)}{s^2}\right) = 0. \end{aligned}$$

This type of behavior is shown in Figure 8.10(*b*).

Example 8.13 (Statistical Modeling Errors Due to Unmodeled System Driving Noise) With the models

Real World	Kalman Filter Model
$\dot{x}_1 = w$	$\dot{x}_1 = 0$
$\dot{x}_2 = x_1$	$\dot{x}_2 = x_1$
$z = x_2 + v$	$z = x_2 + v$

(8.47)

the Kalman filter equations yield the following relationships:

$$\left.\begin{aligned} \dot{p}_{11} &= \frac{-1}{R} p_{12}^2 \\ \dot{p}_{12} &= p_{11} - \frac{p_{12} p_{22}}{R} \\ \dot{p}_{22} &= 2p_{12} - \frac{p_{22}^2}{R} \end{aligned}\right\} \Rightarrow \text{ in the steady state: } \begin{aligned} & p_{12} = 0, \ \overline{K} = PH^{\mathrm{T}} R^{-1} = 0, \\ & p_{11} = 0, \qquad \hat{x}_1 = \text{const}, \\ & p_{22} = 0, \qquad \hat{x}_2 = \text{ramp}. \end{aligned} \tag{8.48}$$

Since x_1 is not constant (due to the driving noise w), the state estimates will not converge to the states, as illustrated in the simulated case plotted in Figure 8.11. This figure shows the behavior of the Kalman gain $\overline{K}_1$, the estimated state component $\hat{x}_1$, and the "true" state component x_1 in a discrete-time simulation of the above model. Because the assumed process noise is zero, the Kalman filter gain $\overline{K}_1$ converges to zero. Because the gain converges to zero, the filter is unable to track the errant state vector component x_1, a random-walk process. Because the filter is unable to track the true state, the innovations (the difference between the predicted

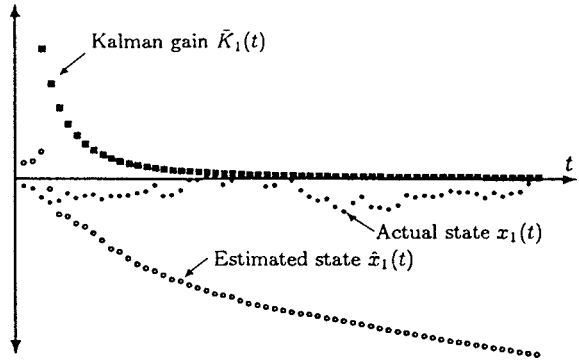


Figure 8.11 Diverging estimation error due to unmodeled process noise.

measurement and the actual measurement) continue to grow without bound. Even though the gains are converging to zero, the product of the gain and the innovations (used in updating the state estimate) can be significant.

In this example, $\sigma^2_{x_1}(t) = \sigma^2_{x_1}(0) + \sigma^2_w t$. That is, the variance of the system state itself diverges. As in the case of unmodeled states, the innovations vector $[z - H\hat{x}]$ will show effects of the "missing" system driving noise.

Example 8.14 (Parametric Modeling Errors) Having the wrong parameters in the system dynamic coefficients F, state transition matrix Φ, or output matrix H can and does bring about nonconvergence of the filter. This type of nonconvergence can be demonstrated by an example with the wrong period of a sinusoid in the output matrix:

Real World	Kalman Filter Model
$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = 0$	$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = 0$
$z = [\sin \Omega t \mid \cos \Omega t] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + v$	$z = [\sin \omega t \mid \cos \omega t] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + v$
$v =$ white noise	$v =$ white noise
No *a priori* information exists	No *a priori* information exists

(8.49)

In this case, the optimum filter is the "least-squares" estimator acting over the measurement interval $(0, T)$. Since this is a continuous case,

$$J = \int_0^T (z - H\hat{x})^T (z - H\hat{x}) dt \tag{8.50}$$

is the performance index being minimized.[2] Its gradient is

$$\frac{\partial J}{\partial \hat{x}} = 0 \quad \Rightarrow \quad 0 = 2\int_0^{\mathrm{T}} H^{\mathrm{T}} z\, dt - 2\int_0^{\mathrm{T}} (H^{\mathrm{T}}H)\hat{x}\, dt, \tag{8.51}$$

where $\hat{x}$ is unknown but constant. Therefore,

$$\hat{x} = \left[\int_0^{\mathrm{T}} H^{\mathrm{T}}H\, dt\right]^{-1}\left[\int_0^{\mathrm{T}} H^{\mathrm{T}} z\, dt\right], \tag{8.52}$$

where

$$H = [\sin \omega t | \cos \omega t]; \qquad z = [\sin \Omega t | \cos \Omega t]\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + v \tag{8.53}$$

and

$$\omega = 2\pi f = \frac{2\pi}{p}, \tag{8.54}$$

where p is the period of the sinusoid. For simplicity, let us choose the sampling time as $T = Np$, an integer multiple of the period, so that

$$\hat{x} = \begin{bmatrix} \dfrac{Np}{2} & 0 \\ 0 & \dfrac{Np}{2} \end{bmatrix}^{-1}\begin{bmatrix} \displaystyle\int_0^{Np} \sin(\omega t) z(t)\, dt \\ \displaystyle\int_0^{Np} \cos(\omega t) z(t)\, dt \end{bmatrix} = \begin{bmatrix} \dfrac{2}{Np}\displaystyle\int_0^{Np} z(t)\sin \omega t\, dt \\ \dfrac{2}{Np}\displaystyle\int_0^{Np} z(t)\cos \omega t\, dt \end{bmatrix}. \tag{8.55}$$

Concentrating on the first component $\hat{x}_1$, one can obtain its solution as

$$\begin{aligned} \hat{x}_1 = &\left\{\frac{2}{Np}\left[\frac{\sin(\omega-\Omega)t}{2(\omega-\Omega)} - \frac{\sin(\omega+\Omega)t}{2(\omega+\Omega)}\right]\Bigg|_{t=0}^{t=Np}\right\}x_1 \\ &+ \left\{\frac{2}{Np}\left[\frac{-\cos(\omega-\Omega)t}{2(\omega-\Omega)} - \frac{\cos(\omega+\Omega)t}{2(\omega+\Omega)}\right]\Bigg|_{t=0}^{t=Np}\right\}x_2 \\ &+ \frac{2}{Np}\int_0^{Np} v(t)\sin \omega t\, dt. \end{aligned}$$

By setting $v = 0$ (ignoring the estimation error due to measurement noise), one obtains the result

$$\begin{aligned} \hat{x}_1 = &\frac{2}{Np}\left[\frac{\sin(\omega-\Omega)Np}{2(\omega-\Omega)} - \frac{\sin(\omega+\Omega)Np}{2(\omega+\Omega)}\right]x_1 \\ &+ \frac{2}{Np}\left[1 - \frac{\cos C(\omega-\Omega)Np}{2(\omega-\Omega)} + \frac{1-\cos(\omega+\Omega)t}{2(\omega+\Omega)}\right]x_2. \end{aligned}$$

[2]See Chapter 1.

For the case where $\omega \to \Omega$,

$$\left.\begin{aligned} \frac{\sin(\omega-\Omega)Np}{2(\omega-\Omega)} &= \frac{Np}{2}\frac{\sin x}{x}\Big|_{x\to 0} = \frac{Np}{2}, \\ \frac{\sin(\omega+\Omega)Np}{2(\omega+\Omega)} &= \frac{\sin[(4\pi/p)Np]}{2\,\Omega} = 0, \\ \frac{1-\cos(\omega-\Omega)Np}{2(\omega-\Omega)} &= \frac{1-\cos x}{x}\Big|_{x\to 0} = 0, \\ \frac{1-\cos(\omega+\Omega)Np}{2(\omega+\Omega)} &= \frac{1-\cos[(4\pi/p)Np]}{2\,\Omega} = 0, \end{aligned}\right\} \tag{8.56}$$

and $\hat{x}_1 = x_1$. In any other case, $\hat{x}_1$ would be a biased estimate of the form

$$\hat{x}_1 = \Upsilon_1 x_1 + \Upsilon_2 x_2, \qquad \text{where} \quad \Upsilon_1 \neq 1, \quad \Upsilon_2 \neq 0. \tag{8.57}$$

Similar behavior occurs with $\hat{x}_2$.

Wrong parameters in the system and/or output matrices may not cause the filter covariance matrix or the state vector to look unusual. However, the innovations vector $[z - H\hat{x}]$ will generally show detectable effects of nonconvergence.

This can be cured only by making sure that the right parameter values are used in the filter model. In the real world, this is often impossible to do precisely, since the right values are not known and can only be estimated. If the amount of degradation obtained is unacceptable, consider letting the questionable parameters become state variables to be estimated by extended (linearized nonlinear) filtering.

Example 8.15 (Parameter Estimation) This reformulation provides an example of a nonlinear estimation problem:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = 0, \qquad x_3 = \Omega. \tag{8.58}$$

Here, something is known about Ω, but it is not known precisely enough. One must choose

$$\hat{x}_3(0) = \text{the "best guess" value of } \Omega,$$

$$P_{33}(0) = \sigma^2_{\tilde{x}_3}(0) = \text{a measure of uncertainty in } \tilde{x}_3(0).$$

Nonlinearities in the real-world system also cause nonconvergence or even divergence of the Kalman estimates.

8.2.5 Analysis and Repair of Covariance Matrices

Covariance matrices must be *nonnegative definite*. By definition, their characteristic values must be nonnegative. However, if any of them are *theoretically* zero—or even close to zero—then there is always the risk that roundoff will cause some roots to

become negative. If roundoff errors should produce an *indefinite* covariance matrix (i.e., one with both positive and negative characteristic values), then there is a way to replace it with a "nearby" nonnegative-definite matrix.

Testing for Positive Definiteness Checks that can be made for the definiteness of a symmetric matrix P include the following:

- If a *diagonal element* $a_{ii} < 0$, then the matrix is *not* positive definite, *but the matrix can have all positive diagonal elements and still fail to be positive definite*.
- If *Cholesky decomposition* $P = CC^{\mathrm{T}}$ fails due to a negative argument in a square root, the matrix is indefinite, or at least close enough to being indefinite that roundoff errors cause the test to fail.
- If *modified Cholesky decomposition* $P = UDU^{\mathrm{T}}$ produces an element $d_{ii} \leq 0$ in the diagonal factor D, then the matrix is not positive definite.
- *Singular value decomposition* yields all the characteristic values and vectors of a symmetric matrix. It is implemented in the MATLAB function `svd`.

The first of these tests is not very reliable unless the dimension of the matrix is 1.

Example 8.16 (3 × 3 Covariance Matrix Examples) The following two 3×3 matrices have positive diagonal values and consistent correlation coefficients, yet neither of them is *positive definite* and the first is actually indefinite:

Matrix	Correlation Matrix	Singular Values
$\begin{bmatrix} 343.341 & 248.836 & 320.379 \\ 248.836 & 336.83 & 370.217 \\ 320.379 & 370.217 & 418.829 \end{bmatrix}$	$\begin{bmatrix} 1. & 0.73172 & 0.844857 \\ 0.73172 & 1. & 0.985672 \\ 0.844857 & 0.985672 & 1. \end{bmatrix}$	$\{1000, 100, -1\}$
$\begin{bmatrix} 343.388 & 248.976 & 320.22 \\ 248.976 & 337.245 & 369.744 \\ 320.22 & 369.744 & 419.367 \end{bmatrix}$	$\begin{bmatrix} 1. & 0.731631 & 0.843837 \\ 0.731631 & 1. & 0.983178 \\ 0.843837 & 0.983178 & 1. \end{bmatrix}$	$\{1000, 100, 0\}$

Repair of Indefinite Covariance Matrices The symmetric eigenvalue–eigenvector decomposition is the more informative test method, because it yields the actual eigenvalues (characteristic values) and their associated eigenvectors (characteristic vectors). The characteristic vectors show the combinations of states with equivalent negative variance. This information allows one to compose a matrix with the identical characteristic vectors but with the offending characteristic values "floored" at zero:

$$P = TDT^{\mathrm{T}} \text{ (symmetric eigenvalue–eigenvector decomposition)}, \tag{8.59}$$

$$D = \mathrm{diag}_i\{d_i\}, \tag{8.60}$$

$$d_1 \geq d_2 \geq d_3 \geq \cdots \geq d_n. \tag{8.61}$$

If

$$d_n < 0, \tag{8.62}$$

then replace P with

$$P^* = TD^*T^{\mathrm{T}}, \tag{8.63}$$

$$D^* = \mathrm{diag}_i\{d_i^*\}, \tag{8.64}$$

$$d_i^* = \begin{cases} d_i & \text{if} \quad d_i \geq 0, \\ 0 & \text{if} \quad d_i < 0. \end{cases} \tag{8.65}$$

8.3 PREFILTERING AND DATA REJECTION METHODS

Prefilters perform data compression of the inputs to Kalman filters. They can be linear continuous, linear discrete, or nonlinear. They are beneficial for several purposes:

1. They allow a discrete Kalman filter to be used on a continuous system without "throwing away" information. For example, the integrate-and-hold prefilter shown in Figure 8.12 integrates over a period T, where T is a sampling time chosen sufficiently small that the dynamic states cannot change significantly between estimates.
2. They attenuate some states to the point where they can be safely ignored (in a suboptimal filter).
3. They can reduce the required iteration rate in a discrete filter, thereby saving computer time [102].
4. They tend to reduce the range of the dynamic variables due to added noise so that the Kalman filter estimate is less degraded by nonlinearities.

8.3.1 Continuous (Analog) Linear Filters

Example 8.17 (Digital Voltmeter Model) Continuous linear filters are usually used for the first three purposes and must be inserted before the sampling process. An example of the continuous linear prefilter is shown in Figure 8.13. An example of such a prefilter is a digital voltmeter (DVM). A DVM is essentially a time-gated averager with a sampler and quantizer.

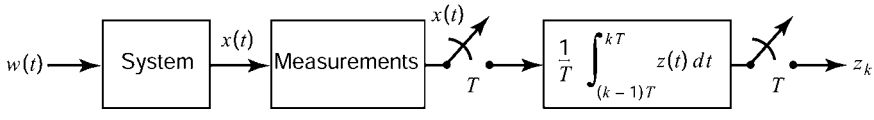


Figure 8.12 Integrate-and-hold prefilter.

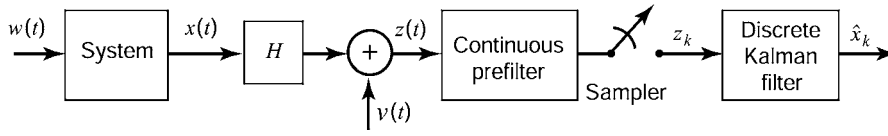


Figure 8.13 Continuous linear prefiltering and sampling.

Thus, the input signal is continuous and the output is discrete. A functional representation is given in Figures 8.14–8.16, where

$$\Delta T = \text{sampling interval}$$
$$\varepsilon = \text{dead time for re-zeroing the integrator}$$
$$\Delta T - \varepsilon = \text{averaging time}$$

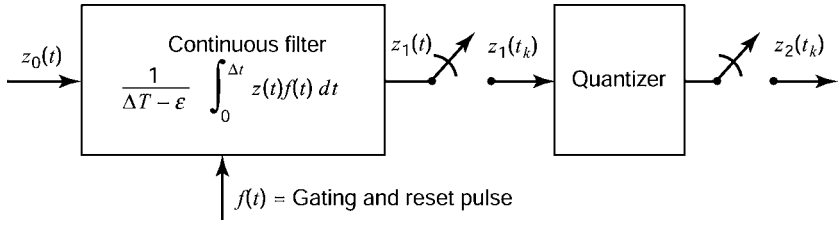


Figure 8.14 Block diagram of a DVM.

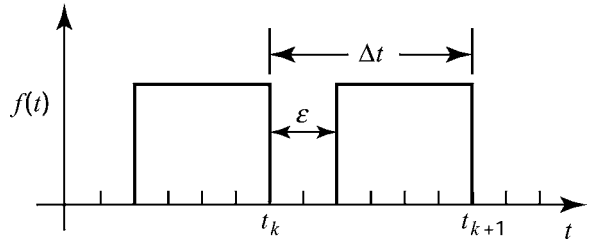


Figure 8.15 DVM gating waveform.

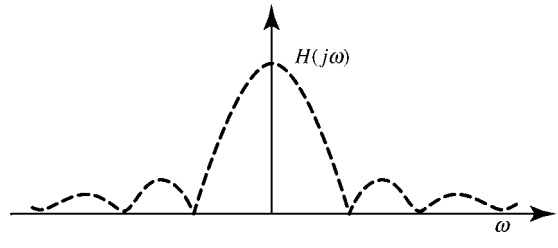


Figure 8.16 DVM frequency response.

and the output is

$$z^1(t_i) = \frac{1}{\Delta T - \varepsilon} \int_{t_i - \Delta T + \varepsilon}^{t_i} z(t)\, dt. \tag{8.66}$$

It can be shown that the frequency response of the DVM is

$$|H(j\omega)| = \left| \frac{\sin \omega[(\Delta T - \varepsilon)/2]}{\omega[(\Delta T - \varepsilon)/2]} \right| \quad \text{and} \quad \theta(j\omega) = -\omega\left(\frac{\Delta T - \varepsilon}{2}\right). \tag{8.67}$$

With *white-noise* continuous input, the output is a white-noise sequence (since the averaging intervals are nonoverlapping). With an *exponentially correlated* random continuous input with correlation time τ_c and autocovariance

$$\psi_z(\tau) = \sigma_z^2 e^{-(|\tau|/\tau_c)}, \tag{8.68}$$

the variance and autocorrelation function of the output random sequence can be shown to be

$$\left.\begin{aligned} u &= \tfrac{\Delta T - \varepsilon}{T_c}, \\ \sigma_{z^1}^2 = \psi_{z^1 z^1}(0) &= f(u)\sigma_z^2, \\ f(u) &= \frac{2}{u^2}(e^{-u} + u - 1) \\ \psi(j-i) &= g(u)\sigma_z^2 e^{-(j-i)\frac{\Delta T}{\tau_c}}, \\ g(u) &= \tfrac{e^u + e^{-u} - 2}{u^2} \end{aligned}\right\}. \tag{8.69}$$

8.3.2 Discrete Linear Filters

These can be used effectively for attenuation of the effects of some of the state variables to the point where they can be safely ignored. (However, the sampling rate input to the filter must be sufficiently high to avoid aliasing.)

Note that discrete filters can be used for the third purpose mentioned above: to reduce the discrete filter iteration rate. The input sampling period can be chosen shorter than the output sampling period. This can greatly reduce the computer (time) load (see Figure 8.17).

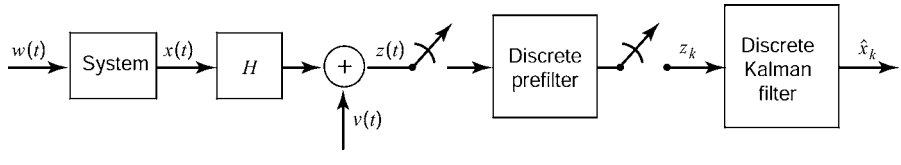


Figure 8.17 Discrete linear prefilter.

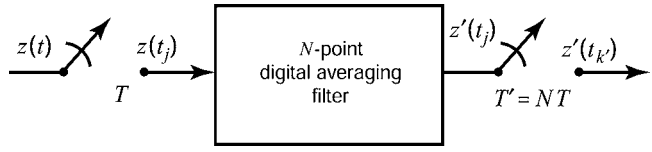


Figure 8.18 Discrete linear averager.

Example 8.18 (Discrete Linear Averager) For the simple digital averager shown in Figure 8.18, let

$$z^1(t_i^1) \equiv \frac{1}{N} \sum_{j=i-N+1}^{i} z(t_j), \quad t_i^1 = iT^1, \tag{8.70}$$

which is the average of N adjacent samples of $z(t_j)$. Note that $z^1(t_i^1)$ and $z^1(t_{i+1}^1)$ use nonoverlapping samples of $z(t_j)$. Then it can be shown that the frequency response is

$$|H(j\omega)| = \frac{|\sin(N\omega T/2)|}{N|\sin(\omega T/2)|}, \; \theta(j\omega) = -\left(\frac{N-1}{2}\right)\omega T. \tag{8.71}$$

If adequate knowledge of the innovations vector $[z - H\hat{x}]$ exists, nonlinear "data rejection filters" can be implemented. Some simple examples are cited below.

Example 8.19 (Data Rejection Filters) For excess amplitude:

$$\text{If } |(z - H\hat{x})_i| > A_{\max}, \text{ then reject data.} \tag{8.72}$$

For excess rate (or change):

$$\text{If } |(z - H\hat{x})_{i+1} - (z - H\hat{x})_i| > \delta A_{\max}, \text{ then reject data.} \tag{8.73}$$

Many other ingenious techniques have been used, but they usually depend on the specifics of the problem.

8.4 STABILITY OF KALMAN FILTERS

The dynamic stability of a system usually refers to the behavior of the state variables, not the estimation errors. This applies as well to the behavior of the homogeneous part of the filter equations. However, the mean-squared estimation errors may remain bounded even if the system is unstable.[3]

If the actual measurement processing in the Kalman filter state equations is neglected, then the resulting equations characterize the stability of the filter itself.

[3]See, for example, Gelb et al. [95], pp. 22, 31, 36, 53, 72, or Maybeck [183], p. 278.

In the continuous case, these equations are

$$\dot{\hat{x}}(t) = [F(t) - \overline{K}(t)H(t)]\hat{x}(t), \tag{8.74}$$

and in the discrete case,

$$\begin{aligned}\hat{x}_k(+) &= \Phi_{k-1}\hat{x}_{k-1}(+) - \overline{K}_k H_k \Phi_{k-1}\hat{x}_{k-1}(+) \\ &= [I - \overline{K}_k H_k]\Phi_{k-1}\hat{x}_{k-1}(+).\end{aligned} \tag{8.75}$$

The solution of the filter equation 8.74 or 8.75 is uniformly asymptotically stable, which means bounded input–bounded output (BIBO) stability. Mathematically, it implies that

$$\lim_{t\to\infty} ||\hat{x}(t)|| = 0 \tag{8.76}$$

or

$$\lim_{k\to\infty} ||\hat{x}_k(+)|| = 0, \tag{8.77}$$

no matter what the initial conditions are. In other words, the filter is uniformly asymptotically stable if the system model is stochastically controllable and observable. See Chapter 4 for the solution of the matrix Riccati equation $P(t)$ or $P_k(+)$ uniformly bounded from above for large t or $\overline{K}$ independent of $P(0)$. Bounded Q, R (above and below) and bounded F (or Φ) will guarantee stochastic controllability and observability.

The most important issues relating to stability are described in the sections on unmodeled effects, finite wordlength, and other errors (Section 8.2).

8.5 SUBOPTIMAL AND REDUCED-ORDER FILTERS

Suboptimal Filters The Kalman filter has a reputation for being robust against certain types of modeling errors, such as those in the assumed values of the statistical parameters R and Q. This reputation is sometimes tested by deliberate simplification of the known (or at least believed) system model. The motive for these actions is usually to reduce implementation complexity by sacrificing some optimality. The result is called a *suboptimal filter.*

8.5.1 Rationale for Suboptimal Filtering

It is often the case that real hardware is nonlinear but, in the filter model, approximated by a linear system. The algorithms developed in Chapters 4–7 will provide suboptimal estimates. These are

1. Kalman filters (linear optimal estimate),
2. linearized Kalman filters, and
3. extended Kalman filters.

Even if there is good reason to believe that the real hardware is truly linear, there may still be reasons to consider suboptimal filters. Where there is doubt about the absolute certainty of the model, there is always a motivation to meddle with it, especially if meddling can decrease the implementation requirements. Optimal filters are generally demanding on computer throughput, and optimality is unachievable if the required computer capacity is not available. Suboptimal filtering can reduce the requirements for computer memory, throughput, and cost. A suboptimal filter design may be "best" if factors other than theoretical filter performance are considered in the trade-offs.

8.5.2 Techniques for Suboptimal Filtering

These techniques can be divided into three categories:

1. modifying the optimal gain $\overline{K}_k$ or $\overline{K}(t)$,
2. modifying the filter model, and
3. other techniques.

Techniques for Evaluating Suboptimal Filters The covariance matrix P may not represent the actual estimation error, and the estimates may be biased. In the following section, the dual-state technique for evaluating performance of linear suboptimal filters will be discussed.

MODIFICATION OF $\overline{K}(t)$ OR $\overline{K}_k$ Consider the system

$$\begin{aligned} \dot{x} &= Fx + Gw, \qquad Q = \text{cov}(w), \\ z &= Hx + v, \qquad R = \text{cov}(v) \end{aligned} \tag{8.78}$$

with state transition matrix Φ.

The state estimation portion of the optimal linear estimation algorithm is then, for the continuous case,

$$\dot{\hat{x}} = F\hat{x} + \overline{K}(t)[z - H\hat{x}] \text{ with } \hat{x}(0) = \hat{x}_0, \tag{8.79}$$

and for the discrete case,

$$\hat{x}_k(+) = \Phi_{k-1}\hat{x}_{k-1}(+) + \overline{K}_k[z_k - H_k(\Phi_{k-1}\hat{x}_{k-1}(+))] \tag{8.80}$$

with initial conditions $\hat{x}_0$.

Schemes for retaining the structure of these algorithms using modified gains include the Wiener filter and approximating functions.

First, consider the *Wiener filter*, which is a useful suboptimal filtering method when the gain vector $\overline{K}(t)$ is time varying but quickly reaches a constant nonzero steady-state value. Typical settling behaviors are shown in Figure 8.19.

A Wiener filter results from the approximation

$$\overline{K}(t) \approx \overline{K}(\infty). \tag{8.81}$$

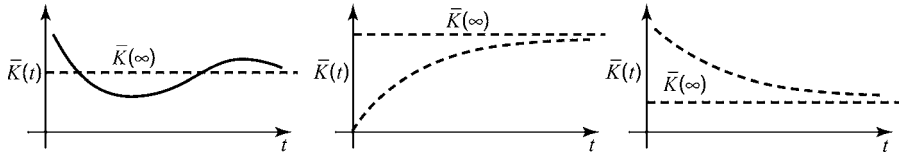


Figure 8.19 Settling of Kalman gains.

If, in addition, the matrices F and H are time invariant, the matrix of transfer functions characterizing the Wiener filter can easily be computed:

$$\dot{\hat{x}} = F\hat{x} + \overline{K}(\infty)[z - H\hat{x}] \tag{8.82}$$

$$\Rightarrow \frac{\hat{x}(s)}{z(s)} = \overline{K}(\infty)[sI - F + \overline{K}(\infty)H]^{-1}. \tag{8.83}$$

The corresponding steady-state sinusoidal frequency response matrix is

$$\frac{|\hat{x}(j\omega)|}{|z(j\omega)|} = |\overline{K}(\infty)[(j\omega)I - F + \overline{K}(\infty)H]^{-1}|. \tag{8.84}$$

Among the advantages of the Wiener filter are that—its structure being identical to that of conventional filters—all the tools of pole-zero, frequency response, and transient response analysis using Laplace transforms can be employed to gain "engineering insight" into the behavior of the filter. Among the disadvantages of this approach is that it cannot be used if $\overline{K}(\infty) \neq$ constant or $\overline{K}(\infty) = 0$. The typical penalty is a poorer transient response (slower convergence) than with optimal gains.

The second scheme for retaining the structure of the algorithms using modified gains is that of *approximating functions*. The optimal time-varying gains $\overline{K}_{\mathrm{OP}}(t)$ are often approximated by simple functions.

For example, one can use piecewise constant approximation, $\overline{K}_{\mathrm{pwc}}$, a piecewise linear approximation, $\overline{K}_{\mathrm{pwl}}$, or a curve fit using smooth functions $\overline{K}_{\mathrm{CF}}$, as shown in Figure 8.20:

$$\overline{K}_{\mathrm{CF}}(t) = C_1 e^{-a_1 t} + C_2(1 - e^{-a_2 t}). \tag{8.85}$$

The advantages of approximating functions over the Wiener filter are that this can handle cases where $\overline{K}(\infty)$ is not constant or $\overline{K}(\infty) = 0$. A result closer to optimal performance is obtained.

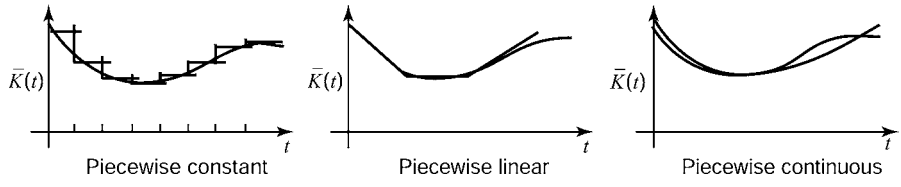


Figure 8.20 Approximating time-varying Kalman gains.

Modification of the Filter Model Let the real-world model (actual system S) be linear,

$$\dot{x}_s = F_s x_s + G_s w_s, \qquad z_s = H_s x_s + v_s.$$

The filter model of the system will, in general, be (intentionally or unintentionally) different:

$$\dot{x}_F = F_F x_F + G_F w_F, \qquad z_F = H_F x_F + v_F.$$

Usually, the intent is to make the filter model less complex than the actual system. This can be done by ignoring some states, prefiltering to attenuate some states, decoupling states, or using frequency domain approximations. Ignoring some states reduces model complexity and provides a suboptimal design. Often, however, little performance degradation occurs.

Example 8.20 (Reduced State Dimension) In this example, one combines two nonobservable states with identical propagation into z.

$$\begin{aligned} \text{From:}\quad & \dot{x}_1 = -ax_1, \\ & \dot{x}_2 = -ax_2, \\ & z = \begin{bmatrix} 2 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + v, \end{aligned} \qquad \begin{aligned} \text{To:}\quad & \dot{x}^1 = -ax^1, \\ & z = x^1 + v, \end{aligned} \tag{8.86}$$

Of course, $x^1 \equiv 2x_1 + 3x_2$ and the *a priori* information must be represented as

$$\begin{aligned} \hat{x}^1(0) &= 2\hat{x}_1(0) + 3\hat{x}_2(0), \\ P_{x^1x^1}(0) &= 4P_{x_1x_1}(0) + 12P_{x_1x_2}(0) + 9P_{x_2x_2}(0). \end{aligned}$$

Example 8.21 (Unmodeled Dynamics) Continuing where Examples 8.10–8.11 left off, one can combine two states if they are "close functions" over the entire measurement interval:

$$\begin{aligned} \text{From:}\quad & \dot{x}_1 = 0, \\ & \dot{x}_2 = 0, \\ & z = [t|\sin t] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + v, \end{aligned} \qquad \begin{aligned} \text{To:}\quad & \dot{x}^1 = 0, \\ & z = tx^1 + v, \end{aligned} \tag{8.87}$$

where the *a priori* information on x^1 must be formulated and where z is available on the interval $(0,\ \pi/20)$.

Example 8.22 (Ignoring Small Effects)

$$\begin{aligned} \text{From:}\quad & \dot{x}_1 = -ax_1, \\ & \dot{x}_2 = -bx_2, \\ & z = x_1 + x_2 + v \text{ on}(0, T), \end{aligned} \qquad \begin{aligned} \text{To:}\quad & \dot{x}_2 = -bx_2, \\ & z = x_2 + v, \end{aligned} \tag{8.88}$$

with

$$E(x_1^2) = 0.1, \quad E(x_2^2) = 10.0, \qquad \text{desired} \quad P_{22}(T) = 1.0. \tag{8.89}$$

Example 8.23 (Relying on Time Orthogonality of States)

From:	To:
$\dot{x}_1 = 0,$	
$\dot{x}_2 = 0,$	$\dot{x}_2 = 0,$
$z = [1 \mid \sin t]\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + v,$	$z = (\sin t)x_2 + v,$
z available on $(0, T)$ with T large	
$P_{22}(0) = \text{large}.$	

(8.90)

Prefiltering to Simplify the Model A second technique for modifying the filter model of the real world is to ignore states after prefiltering to provide attenuation. This is illustrated in Figure 8.21.

Of course, prefiltering has deleterious side effects, too. It may require other changes to compensate for the measurement noise v becoming time correlated after passing through the prefilter and to account for some distortion of those states in the passband of the prefilter (e.g., amplitude and phase of sinusoidals and wave

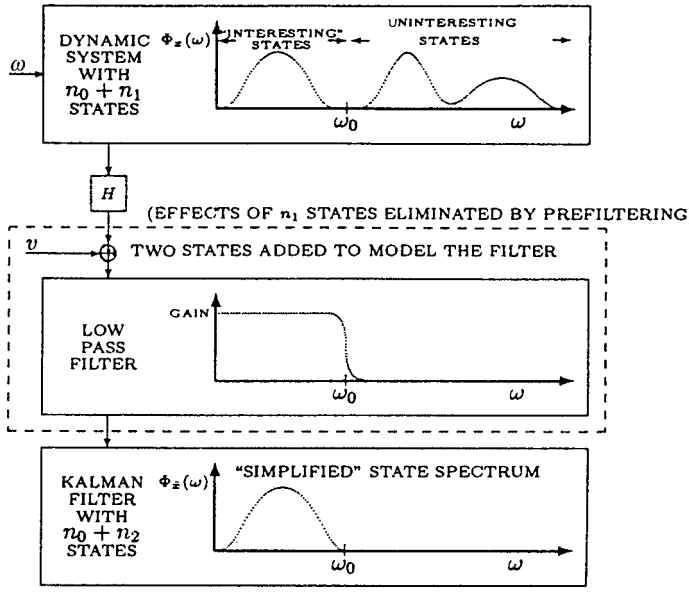


Figure 8.21 Low-pass prefiltering for model simplification.

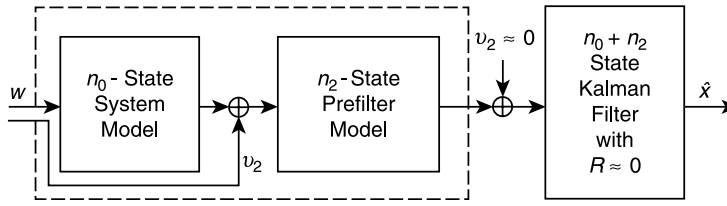


Figure 8.22 Modified system model for low-pass prefiltering.

shape of signals). The filter may be modified as shown in Figure 8.22 to compensate for such effects. Hopefully, the net result is a smaller filter than before.

Example 8.24 (Weakly Coupled Dynamics) Consider the second-order system with coupling coefficient τ, represented by the equations

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -\alpha_1 & 0 \\ \tau & -\alpha_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} w_1 \\ w_2 \end{bmatrix},$$
$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}.$$

If τ is sufficiently small, one can treat the system as two separate uncoupled systems with two separate corresponding Kalman filters:

$$\begin{aligned} \dot{x}_1 &= -\alpha_1 x_1 + w_1, \quad & \dot{x}_2 &= -\alpha_2 x_2 + w_2, \\ z_1 &= x_1 + v_1, & z_2 &= x_2 + v_2. \end{aligned} \tag{8.91}$$

The advantage of this decoupling method is a large reduction in computer load. However, the disadvantage is that the estimation error has increased variance and can be biased.

Frequency-Domain Approximations These can be used to formulate a suboptimal filter for a system of the sort

$$\dot{x} = Fx + Gw, \qquad z = Hx + v.$$

Often, some of the states are stationary random processes whose PSDs can be approximated as shown in Figure 8.23.

A filter designed for the approximate spectrum may well use fewer states.

Example 8.25 (Noise Covariance Uncertainty) Sometimes the general structure of the random process model is known, but the parameters are not known precisely:

$$\dot{x} = -\alpha x + w \text{ and } \sigma_w \text{ are uncertain} \tag{8.92}$$

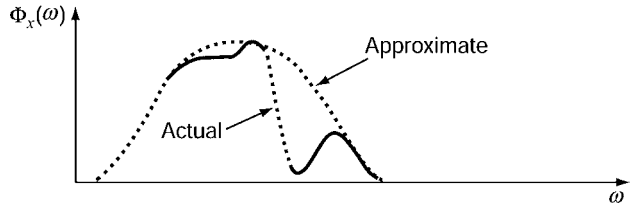


Figure 8.23 Frequency-domain approximation.

Replacing this model by a random walk

$$\dot{x} = w \tag{8.93}$$

in conjunction with *sensitivity studies* will often allow a judicious choice of σ_w with small sensitivity to α uncertainties and small degradation in filter performance.

Example 8.26 (White-Noise Approximation of Broadband Noise) If the system-driving noise and measurement noise are broadband but with sufficiently flat PSDs, they can be replaced by white-noise approximations, as illustrated in Figure 8.24.

Least-Squares Filters Among the other techniques used in practice to intentionally suboptimize linear filters is least-squares estimation. It is equivalent to Kalman filtering if there are no state dynamics ($F = 0$ and $Q = 0$) and is often considered as a candidate for a suboptimal filter if the influence of the actual values of Q and F (or Φ) on the values of the Kalman gain is small.

Observer Methods These simpler filters can be designed by choosing the eigenvalues of a filter of special structure. The design of suboptimal filters using engineering insight is often possible. Ingenuity based on physical insights with regard to design of suboptimal filters is considered an art, not a science, by some practitioners.

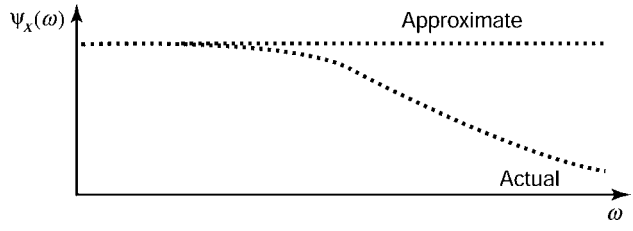


Figure 8.24 White-noise approximation of broadband noise.

8.5.3 Dual-State Evaluation of Suboptimal Filters

Dual-State Analysis This form of analysis takes its name from the existence of two views of reality:

1. The so-called *system model* (or *truth model*) of the actual system under study. This model is used to generate the observations input to the suboptimal filter. It should be a reasonably complete model of the actual system under consideration, including all known phenomena that are likely to influence the performance of the estimator.
2. The *filter model*, which is a reduced-order version of the system model, is usually constrained to contain all the states in the *domain* of the measurement sensitivity matrix (i.e., the states that contribute to the measurement) and possibly other state components of the system model as well. The filter model is to be used by a proposed filter implementation with significantly reduced computational complexity but (hopefully) not greatly reduced fidelity. The performance of the reduced-order filter implementation is usually measured by how well its estimates agree with those of the actual system state. It may not estimate *all* of the components of the state vector of the system model, however. In that case, the evaluation of its estimation accuracy is restricted to the common state vector components.

Performance analysis of suboptimal filter errors requires the simultaneous consideration of both (system and filter) state vectors, which are combined into one *dual-state vector*.

NOTATION Let us use a superscript notation to distinguish between these models. A superscript S will denote the *system model*, and a superscript F will denote the *filter model*, as illustrated later in Figure 8.30.[4] There are two commonly used definitions for the dual-state vector:

1. $x_k^{\text{dual}} = \begin{bmatrix} x_k^S \\ \hat{x}_k^F(+) \end{bmatrix}$ (concatenation of the two vectors) and
2. $\tilde{x}_k^{\text{dual}} = \begin{bmatrix} \tilde{x}_k^S(+) \\ x_k^S \end{bmatrix}$, $\tilde{x}_k^S(+) = \hat{x}_k^F(+) - x_k^S$.

In the second definition, the filter state vector x^F may have lower dimension than the system state vector x^S—due to neglected state components of the system in the suboptimal filter model. In that case, the missing components in x^F can be padded with zeros to make the dimensions match.

[4]The system model is called the *truth* model in the figure. That nomenclature here would lead us to use a superscript T, however, and that notation is already in use to denote transposition.

In dual-state analysis for evaluating suboptimal filters, let the following definitions apply:

Actual System	Filter Model	
$\dot{x}^S = F_S x^S + G_S w^S$	$\dot{x}^F = F_F x^F + G_F w^F$	(8.94)
$z^S = H_S x^S + v^S$	$z^F = H_F x^F + v^F$	
$z^F = z^S$	$H_F = H_S,\ v^F = v^S$	(8.95)
$\eta_k^S \equiv \int_{t_{k-1}}^{t_k} \Phi_S(t_k - \tau) G_S(\tau) w^S(\tau)\, d\tau$, $\hat{x}_k^F(+) = \Phi_F \hat{x}_{k-1}^F(+) + \overline{K}_k[z_k^F - H_F \Phi_F \hat{x}_{k-1}^F(+)]$, $\Phi_S \equiv$ system state transition matrix, $\Phi_F \equiv$ state transition matrix of filter model, $Q_S \equiv \text{cov}(w^S)$, $Q_F \equiv \text{cov}(w^F)$,		(8.96)

Let $\Gamma_k \equiv (I - \overline{K}_k H_F)$, and let

$$A \equiv \begin{bmatrix} \Phi_F & \Phi_F - \Phi_S \\ 0 & \Phi_S \end{bmatrix}, \tag{8.97}$$

$$B \equiv \begin{bmatrix} \Gamma_k & 0 \\ 0 & I \end{bmatrix}. \tag{8.98}$$

Let the prediction estimation error be

$$\tilde{x}_k^S(-) \equiv \hat{x}_k^F(-) - x_k^S \tag{8.99}$$

and let the filtered estimation error be

$$\tilde{x}_{k-1}^S(+) \equiv \hat{x}_{k-1}^F(+) - x_{k-1}^S. \tag{8.100}$$

Then the prediction error equation is

$$\begin{bmatrix} \tilde{x}_k^S(-) \\ x_k^S \end{bmatrix} = A \begin{bmatrix} \tilde{x}_{k-1}^S(+) \\ x_{k-1}^S \end{bmatrix} + \begin{bmatrix} -\eta_{k-1}^S \\ \eta_{k-1}^S \end{bmatrix} \tag{8.101}$$

and the filtered error equation is

$$\begin{bmatrix} \tilde{x}_{k-1}^S(+) \\ x_{k-1}^S \end{bmatrix} = B \begin{bmatrix} \tilde{x}_{k-1}^S(-) \\ x_{k-1}^S \end{bmatrix} + \begin{bmatrix} \overline{K}_{k-1} v_{k-1}^S \\ 0 \end{bmatrix}. \tag{8.102}$$

Taking the expected value E and the covariance of Equations 8.101 and 8.102 yields the following recursive relationships:

$$\left.\begin{aligned} E_{\text{cov}} &= A \cdot P_{\text{cov}} \cdot A^{\text{T}} + \text{cov}L \\ P_{\text{cov}} &= B \cdot E_{\text{cov}} \cdot B^{\text{T}} + \text{cov}P \end{aligned}\right\} \text{(covariance propagation)}, \tag{8.103}$$

$$\left.\begin{aligned} E_{\text{DX}} &= A \cdot P_{\text{DX}} \\ P_{\text{DX}} &= B \cdot E_{\text{DX}} \end{aligned}\right\} \text{(bias propagation)}, \tag{8.104}$$

where the newly introduced symbols are defined as follows:

$$\begin{aligned} E_{\text{cov}} &\equiv \text{cov}\begin{bmatrix} \tilde{x}_k^S(-) \\ x_k^S \end{bmatrix} \text{(predicted dual-state vector)}, \\ \text{cov}L &\equiv \text{cov}\begin{bmatrix} -\eta_{k-1}^S \\ \eta_{k-1}^S \end{bmatrix}, \\ P_{\text{cov}} &\equiv \text{cov}\begin{bmatrix} \tilde{x}_{k-1}^S(+) \\ x_{k-1}^S \end{bmatrix} \text{(filtered covariance)}, \\ \text{cov}P &\equiv \text{cov}\begin{bmatrix} \overline{K}_{k-1}\nu_{k-1}^S \\ 0 \end{bmatrix}, \\ &= \begin{bmatrix} \overline{K}_{k-1}\text{cov}(\nu_{k-1}^S)\overline{K}_{k-1}^{\text{T}} & 0 \\ 0 & 0 \end{bmatrix}, \\ P_{\text{DX}} &= E\begin{bmatrix} \tilde{x}_{k-1}^S(+) \\ x_{k-1}^S \end{bmatrix} \text{(expected value of filtered dual-state vector)}, \\ E_{\text{DX}} &= E\begin{bmatrix} \tilde{x}_k^S(-) \\ x_k^S \end{bmatrix} \text{(expected value of predicted dual-state vector)}. \end{aligned} \tag{8.105}$$

The suboptimal estimate $\hat{x}$ can be biased. The estimation error covariance can depend on the system state covariance. To show this, one can use the dual-state equations. The bias propagation equations

$$E_{\text{DX}} = A \cdot P_{\text{DX}}, P_{\text{DX}} = B \cdot E_{\text{DX}}$$

become

$$E\begin{bmatrix} \tilde{x}_k^S(-) \\ x_k^S \end{bmatrix} = AE\begin{bmatrix} \tilde{x}_{k-1}^S(+) \\ x_{k-1}^S \end{bmatrix} \tag{8.106}$$

and

$$E\begin{bmatrix} \tilde{x}^S_{k-1}(+) \\ x^S_{k-1} \end{bmatrix} = BE\begin{bmatrix} \tilde{x}^S_{k-1}(-) \\ x^S_{k-1} \end{bmatrix}. \tag{8.107}$$

Clearly, if $E[\tilde{x}^S_k(+)] \neq 0$, then the estimate becomes biased. If

$$\Phi_F \neq \Phi_S \tag{8.108}$$

and

$$E(x^S) \neq 0, \tag{8.109}$$

which often is the case (e.g., x^S may be a deterministic variable, so that $E(x^S) = x^S \neq 0$, x^S may be a random variable with nonzero mean). Similarly, an examination of the covariance propagation equations

$$E_{\text{cov}} = A(P_{\text{cov}})A^{\text{T}} + \text{cov}L$$
$$P_{\text{cov}} = B(E_{\text{cov}})B^{\text{T}} + \text{cov}P$$

shows that $\text{cov}[\tilde{x}^S_k(-)]$ depends on $\text{cov}(x^S_k)$. (See Problem 8.5.)

If

$$\Phi_F \neq \Phi_S \tag{8.110}$$

and

$$\text{cov}(x^S) \neq 0, \tag{8.111}$$

which often is the case with the suboptimal filter, the estimate $\hat{x}$ is unbiased and the estimation error covariance is independent of the system state.

8.6 SCHMIDT–KALMAN FILTERING

8.6.1 Historical Background

Stanley F. Schmidt was an early and successful advocate of Kalman filtering. He was working at the NASA Ames Laboratory in Mountain View, California, when Kalman presented his results there in 1959. Schmidt immediately began applying it to a problem then under study at Ames, which was the space navigation problem (i.e., trajectory estimation) for the upcoming Apollo project for manned exploration of the moon. (In the process, Schmidt discovered what is now called *extended* Kalman filtering.) Schmidt was so impressed with his results that he set about proselytizing his professional colleagues and peers to try Kalman filtering.

Schmidt also derived and evaluated many practical methods for improving the numerical stability of the procedures used and for reducing the computational requirements of Kalman filtering. Many of these results were published in journal articles,

technical reports, and books. In [239] Schmidt presents an approach (now called *Schmidt—Kalman filtering*) for reducing the computational complexity of Kalman filters by eliminating some of the computation for "nuisance variables," which are state variables that are of no interest for the problem at hand—except that they are part of the system state vector.

Schmidt's approach is suboptimal in that it sacrifices estimation performance for computational performance. It enabled Kalman filters to be approximated so that they could be implemented in real time on the computers of that era (the mid-1960s). However, it still finds useful application today for implementing Kalman filters on small embedded microprocessors.

The types of nuisance variables that find their way into the Kalman filter state vector include those used for modeling correlated measurement noise (e.g., colored, pastel, or random-walk noise). We generally have no interest in the memory state of such noise. We just want to filter it out.

Because the dynamics of measurement noise are generally *not* linked to the other system state variables, these added state variables are not dynamically coupled to the other state variables. That is, the elements in the dynamic coefficient matrix linking the two state variable types (states related to correlated measurement noise and states not related to correlated measurement noise) are zero. In other words, if the ith state variable is of one type and the jth state variable is of the other type, then the element f_{ij} in the ith row and jth column of the dynamic coefficient matrix F will always be zero.

Schmidt was able to take advantage of this, because it means that the state variables can be reordered in the state vector such that the nuisance variables appear last. The resulting dynamic equation then has the form

$$\frac{d}{dt}\mathbf{x}(t) = \begin{bmatrix} F_{\varepsilon}(t) & 0 \\ 0 & F_{v}(t) \end{bmatrix}\mathbf{x}(t) + \mathbf{w}(t) \tag{8.112}$$

such that F_v represents the dynamics of the nuisance variables and F_ε represents the dynamics of the other state variables.

It is this partitioning of the state vector that leads to the reduced-order, suboptimal filter called the Schmidt–Kalman filter.

8.6.2 Derivation

Partitioning the Model Equations Let[5]

$n = n_\varepsilon + n_v$ be the total number of state variables,

n_ε be the number of essential variables, whose values are of interest for the application, and

[5]This derivation follows that in [50].

n_v be the number of nuisance variables, whose values are of no intrinsic interest and whose dynamics are not coupled with those of the essential state variables.

Then the state variables can be reordered in the state vector such that the essential variables precede the nuisance variables:

$$\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{n_\varepsilon} \\ x_{n_\varepsilon+1} \\ x_{n_\varepsilon+2} \\ x_{n_\varepsilon+3} \\ \vdots \\ x_{n_\varepsilon+n_v} \end{bmatrix} \begin{matrix} \left.\begin{matrix} \\ \\ \\ \\ \end{matrix}\right\} \text{essential variables} \\ \left.\begin{matrix} \\ \\ \\ \\ \end{matrix}\right\} \text{nuisance variables} \end{matrix} \tag{8.113}$$

$$= \left[\frac{\mathbf{x}_\varepsilon}{\mathbf{x}_v}\right], \tag{8.114}$$

where the state vector has been partitioned into a subvector $\mathbf{x}_\varepsilon$ of the essential state variables and a subvector $\mathbf{x}_v$ of nuisance variables.

Partitioning of State Dynamic Models We know that these two state variable types are not linked dynamically, so that the system dynamic model has the form

$$\frac{d}{dt}\left[\frac{\mathbf{x}_\varepsilon(t)}{\mathbf{x}_v(t)}\right] = \left[\begin{array}{c|c} F_\varepsilon(t) & 0 \\ \hline 0 & F_v(t) \end{array}\right]\left[\frac{\mathbf{x}_\varepsilon(t)}{\mathbf{x}_v(t)}\right] + \left[\frac{\mathbf{w}_\varepsilon(t)}{\mathbf{w}_v(t)}\right] \tag{8.115}$$

in continuous time, where the process noise vectors $\mathbf{w}_\varepsilon$ and $\mathbf{w}_v$ are uncorrelated. That is, the covariance matrix of process noise

$$Q = \left[\begin{array}{c|c} Q_{\varepsilon\varepsilon} & 0 \\ \hline 0 & Q_{vv} \end{array}\right] \tag{8.116}$$

for the continuous-time model (as well as for the discrete-time model). That is, the cross-covariance block $Q\varepsilon v = 0$.

Partitioned Covariance Matrix The covariance matrix of estimation uncertainty (the dependent variable of the Riccati equation) can also be partitioned as

$$P = \left[\begin{array}{c|c} P_{\varepsilon\varepsilon} & P_{\varepsilon v} \\ \hline P_{v\varepsilon} & P_{vv} \end{array}\right], \tag{8.117}$$

where

the block $P_{\varepsilon\varepsilon}$ is dimensioned $n_\varepsilon \times n_\varepsilon$,
the block $P_{\varepsilon v}$ is dimensioned $n_\varepsilon \times n_v$,
the block $P_{v\varepsilon}$ is dimensioned $n_v \times n_\varepsilon$, and
the block P_{vv} is dimensioned $n_v \times n_v$.

Temporal Covariance Update in Discrete Time The corresponding state transition matrix for the discrete-time model will then be of the form

$$\Phi_k = \left[\begin{array}{c|c} \Phi_{\varepsilon k} & 0 \\ \hline 0 & \Phi_{vk} \end{array}\right], \tag{8.118}$$

$$\Phi_{\varepsilon k} = \exp\left(\int_{t_k}^{t_{k+1}} F_\varepsilon(t)\,dt\right), \tag{8.119}$$

$$\Phi_{vk} = \exp\left(\int_{t_k}^{t_{k+1}} F_v(t)\,dt\right), \tag{8.120}$$

and the temporal update of P will have the partitioned form

$$\begin{aligned} &\left[\begin{array}{c|c} P_{\varepsilon\varepsilon k+1-} & P_{\varepsilon v k+1-} \\ \hline P_{v\varepsilon k+1-} & P_{vvk+1-} \end{array}\right] \\ &\quad = \left[\begin{array}{c|c} \Phi_{\varepsilon k} & 0 \\ \hline 0 & \Phi_{vk} \end{array}\right]\left[\begin{array}{c|c} P_{\varepsilon\varepsilon k+} & P_{\varepsilon v k+} \\ \hline P_{v\varepsilon k+} & P_{vvk+} \end{array}\right]\left[\begin{array}{c|c} \Phi_{\varepsilon k}^{\mathrm{T}} & 0 \\ \hline 0 & \Phi_{vk}^{\mathrm{T}} \end{array}\right] \\ &\qquad + \left[\begin{array}{c|c} Q_{\varepsilon\varepsilon} & 0 \\ \hline 0 & Q_{vv} \end{array}\right], \end{aligned} \tag{8.121}$$

or, in terms of the individual blocks,

$$P_{\varepsilon\varepsilon k+1-} = \Phi_{\varepsilon k} P_{\varepsilon\varepsilon k+} \Phi_{\varepsilon k}^{\mathrm{T}} + Q_{\varepsilon\varepsilon}, \tag{8.122}$$

$$P_{\varepsilon v k+1-} = \Phi_{\varepsilon k} P_{\varepsilon v k+} \Phi_{vk}^{\mathrm{T}}, \tag{8.123}$$

$$P_{v\varepsilon k+1-} = \Phi_{vk} P_{v\varepsilon k+} \Phi_{\varepsilon k}^{\mathrm{T}}, \tag{8.124}$$

$$P_{vvk+1-} = \Phi_{vk} P_{vvk+} \Phi_{vk}^{\mathrm{T}} + Q_{vv}. \tag{8.125}$$

Partitioned Measurement Densitivity Matrix With this partitioning of the state vector, the measurement model will have the form

$$z = [H_\varepsilon \quad | \quad H_v]\left[\frac{\mathbf{x}_\varepsilon(t)}{\mathbf{x}_v(t)}\right] + \mathbf{v} \tag{8.126}$$

$$= \underbrace{H_\varepsilon \mathbf{x}_\varepsilon}_{\substack{\text{essential}\\ \text{state}\\ \text{dependence}}} + \underbrace{H_v \mathbf{x}_v}_{\substack{\text{correlated}\\ \text{noise}}} + \underbrace{\mathbf{v}}_{\substack{\text{uncorrelated}\\ \text{noise}}} . \tag{8.127}$$

8.6.2.1 Schmidt–Kalman Gain

Kalman Gain The Schmidt–Kalman filter does not use the Kalman gain matrix. However, we need to write out its definition in partitioned form to show how its modification results in the Schmidt–Kalman gain.

The Kalman gain matrix would be partitionable conformably, such that

$$\overline{K} = \left[\frac{K_\varepsilon}{K_v}\right] \tag{8.128}$$

$$= \left[\begin{array}{c|c} P_{\varepsilon\varepsilon} & P_{\varepsilon v} \\ \hline P_{v\varepsilon} & P_{vv} \end{array}\right]\left[\frac{H_\varepsilon^{\mathrm{T}}}{H_v^{\mathrm{T}}}\right]$$

$$\left\{[H_\varepsilon \quad | \quad H_v]\left[\begin{array}{c|c} P_{\varepsilon\varepsilon} & P_{\varepsilon v} \\ \hline P_{v\varepsilon} & P_{vv} \end{array}\right]\left[\frac{H_\varepsilon^{\mathrm{T}}}{H_v^{\mathrm{T}}}\right] + R\right\}^{-1} \tag{8.129}$$

and the individual blocks

$$K_\varepsilon = \{P_{\varepsilon\varepsilon} H_\varepsilon^{\mathrm{T}} + P_{\varepsilon v} H_v^{\mathrm{T}}\}\mathcal{C}, \tag{8.130}$$

$$K_v = \{P_{v\varepsilon} H_\varepsilon^{\mathrm{T}} + P_{vv} H_v^{\mathrm{T}}\}\mathcal{C} \tag{8.131}$$

where the common factor

$$\mathcal{C} = \left\{[H_\varepsilon \quad | \quad H_v]\begin{bmatrix} P_{\varepsilon\varepsilon} & P_{\varepsilon v} \\ P_{v\varepsilon} & P_{vv} \end{bmatrix}\left[\frac{H_\varepsilon^{\mathrm{T}}}{H_v^{\mathrm{T}}}\right] + R\right\}^{-1} \tag{8.132}$$

$$= \{H_\varepsilon P_{\varepsilon\varepsilon} H_\varepsilon^{\mathrm{T}} + H_\varepsilon P_{\varepsilon v} H_v^{\mathrm{T}} + H_v P_{v\varepsilon} H_\varepsilon^{\mathrm{T}} + H_v P_{vv} H_v^{\mathrm{T}} + R\}^{-1}. \tag{8.133}$$

However, the Schmidt–Kalman filter will, in effect, force K_v to be zero and redefine the upper block (no longer the K_ε of the Kalman filter) to be optimal under that constraint.

Suboptimal Approach The approach will be to define a suboptimal filter that does not estimate the nuisance state variables but does keep track of the influence they will have on the gains applied to the other state variables.

The suboptimal gain matrix for the Schmidt–Kalman filter has the form

$$\overline{K}_{\text{suboptimal}} = \begin{bmatrix} K_{\text{SK}} \\ 0 \end{bmatrix}, \tag{8.134}$$

where K_{SK} is the $n_\varepsilon \times \ell$ Schmidt–Kalman gain matrix.

This suboptimal filter effectively ignores the nuisance states.

However, the calculation of the covariance matrix P used in defining the gain K_{SK} must still take into account the effect that this constraint has on the state estimation uncertainties and must optimize K_{SK} for that purpose. Here, K_{SK} will effectively be optimal for the constraint that the nuisance states are not estimated. However, the filter will still be *sub*optimal in the sense that filter performance using both Kalman gain blocks (K_ε and K_v) would be superior to that with K_{SK} alone.

The approach still propagates the full covariance matrix P, but the observational update equations are changed to reflect the fact that (in effect) $K_v = 0$.

Suboptimal Observational Update The observational update equation for arbitrary gain $\overline{K}_k$ can be represented in the form

$$P_k(+) = (I_n - \overline{K}_k H_k) P_k(-) (I_n - \overline{K}_k H_k)^{\text{T}} + \overline{K} R \overline{K}^{\text{T}}, \tag{8.135}$$

where n is the dimension of the state vector, I_n is the $n \times n$ identity matrix, ℓ is the dimension of the measurement, H_k is the $\ell \times n$ measurement sensitivity matrix, and R_k is the $\ell \times \ell$ covariance matrix of uncorrelated measurement noise.

In the case where the suboptimal gain K_k has the partitioned form shown in Equation 8.134, the partitioned observational update equation for P will be

$$\begin{aligned} \begin{bmatrix} P_{\varepsilon\varepsilon,k+} & P_{\varepsilon v,k+} \\ P_{v\varepsilon,k+} & P_{vv,k+} \end{bmatrix} = & \left(\begin{bmatrix} I_{n_\varepsilon} & 0 \\ 0 & I_{n_v} \end{bmatrix} - \begin{bmatrix} K_{\text{SK},k} \\ \hline 0 \end{bmatrix} [H_{\varepsilon,k} \quad | \quad H_{v,k}] \right) \\ & \times \begin{bmatrix} P_{\varepsilon\varepsilon,k-} & P_{\varepsilon v,k-} \\ P_{v\varepsilon,k-} & P_{vv,k-} \end{bmatrix} \\ & \times \left(\begin{bmatrix} I_{n_\varepsilon} & 0 \\ 0 & I_{n_v} \end{bmatrix} - \begin{bmatrix} K_{\text{SK},k} \\ \hline 0 \end{bmatrix} [H_{\varepsilon,k} \quad H_{v,k}] \right)^{\text{T}} \\ & + \begin{bmatrix} K_{\text{SK},k} a \\ \hline 0 \end{bmatrix} R_k [K_{\text{SK},k}^{\text{T}} \quad 0]. \end{aligned} \tag{8.136}$$

The summed terms in parentheses can be combined into the following form for expansion:

$$\begin{aligned} &= \begin{bmatrix} I_{n_\varepsilon} - K_{\mathrm{SK},k} H_{\varepsilon,k} & -K_{\mathrm{SK},k} H_{v,k} \\ 0 & I_{n_v} \end{bmatrix} \times \begin{bmatrix} P_{\varepsilon\varepsilon,k-} & P_{\varepsilon v,k-} \\ P_{v\varepsilon,k-} & P_{vv,k-} \end{bmatrix} \\ &\quad \times \begin{bmatrix} I_{n_\varepsilon} - H_{\varepsilon,k}^{\mathrm{T}} K_{\mathrm{SK},k}^{\mathrm{T}} & 0 \\ -H_{v,k}^{\mathrm{T}} K_{\mathrm{SK},k}^{\mathrm{T}} & I_{n_v} \end{bmatrix} + \begin{bmatrix} K_{\mathrm{SK},k} R_k K_{\mathrm{SK},k}^{\mathrm{T}} & 0 \\ 0 & 0 \end{bmatrix}, \end{aligned} \tag{8.137}$$

which can then be expanded to yield the following formulas for the blocks of P (with annotation showing intermediate results that can be reused to reduce the computation):

$$\begin{aligned} P_{\varepsilon\varepsilon,k+} = {} & \underbrace{(I_{n_\varepsilon} - K_{\mathrm{SK},k} H_{\varepsilon,k})}_{\mathcal{A}} P_{\varepsilon\varepsilon,k-} \underbrace{(I_{n_\varepsilon} - K_{\mathrm{SK},k} H_{\varepsilon,k})^{\mathrm{T}}}_{\mathcal{A}^{\mathrm{T}}} \\ & - \overbrace{\underbrace{(I_{n_\varepsilon} - K_{\mathrm{SK},k} H_{\varepsilon,k})}_{\mathcal{A}} P_{v\varepsilon,k-} H_{v,k}^{\mathrm{T}} K_{\mathrm{SK},k}^{\mathrm{T}}}^{\mathcal{B}} \\ & - \overbrace{K_{\mathrm{SK},k} H_{v,k} P_{\varepsilon v,k-} (I_{n_\varepsilon} - K_{\mathrm{SK},k} H_{\varepsilon,k})^{\mathrm{T}}}^{\mathcal{B}^{\mathrm{T}}} \\ & + K_{\mathrm{SK},k} R_k K_{\mathrm{SK},k}^{\mathrm{T}}, \end{aligned} \tag{8.138}$$

$$P_{\varepsilon v,k+} = \overbrace{(I_{n_\varepsilon} - K_{\mathrm{SK},k} H_{\varepsilon,k})}^{\mathcal{A}} P_{\varepsilon v,k-} - K_{\mathrm{SK},k} H_{v,k} P_{vv,k-}, \tag{8.139}$$

$$P_{v\varepsilon,k+} = P_{\varepsilon v,k+}^{\mathrm{T}}, \tag{8.140}$$

$$P_{vv,k+} = P_{vv,k-}. \tag{8.141}$$

Note that P_{vv} is unchanged by the observational update because $\mathbf{x}_v$ is not updated.

This Completes the Derivation of the Schmidt–Kalman Filter The temporal update of P in the Schmidt–Kalman filter will be the same as for the Kalman filter. This happens because the temporal update only models the propagation of the state variables, and the propagation model is the same in both cases.

8.6.3 Implementation Equations

We can now summarize the essential equations from the derivation above, as listed in Table 8.1. These have been rearranged slightly to reuse intermediate results.

TABLE 8.1 Implementation Equations of the Schmidt–Kalman Filter

Observational update

$$\begin{aligned}
\mathcal{C} &= \{H_{\varepsilon k}(P_{\varepsilon\varepsilon k-}\ H_{\varepsilon k}^{\mathrm{T}} + P_{\varepsilon v k-}\ H_{vk}^{\mathrm{T}}) \\
&\quad + H_{vk}(P_{v\varepsilon k-}\ H_{\varepsilon k}^{\mathrm{T}} + P_{vvk-}\ H_{vk}^{\mathrm{T}}) + R_k\}^{-1} \\
K_{\mathrm{SK},k} &= \{P_{\varepsilon\varepsilon k-}\ H_{\varepsilon k}^{\mathrm{T}} + P_{\varepsilon v k-}\ H_{vk}^{\mathrm{T}}\}\mathcal{C} \\
\mathbf{x}_{\varepsilon,k+} &= \mathbf{x}_{\varepsilon,k-} + K_{\mathrm{S-K},k}\{\mathbf{z}_k - H_{\varepsilon k}\mathbf{x}_{\varepsilon,k-}\} \\
\mathcal{A} &= I_{n_\varepsilon} - K_{\mathrm{SK},k}H_{\varepsilon,k} \\
\mathcal{B} &= \mathcal{A}P_{v\varepsilon,k-}\ H_{v,k}^{\mathrm{T}}K_{\mathrm{SK},k}^{\mathrm{T}} \\
P_{\varepsilon\varepsilon,k+} &= \mathcal{A}P_{\varepsilon\varepsilon,k-}\ \mathcal{A}^{\mathrm{T}} - \mathcal{B} - \mathcal{B}^{\mathrm{T}} + K_{\mathrm{SK},k}R_kK_{\mathrm{SK},k}^{\mathrm{T}} \\
P_{\varepsilon v,k+} &= \mathcal{A}P_{\varepsilon v,k-} - K_{\mathrm{SK},k}H_{v,k}P_{vv,k-} \\
P_{v\varepsilon,k+} &= P_{\varepsilon v,k+}^{\mathrm{T}} \\
P_{vv,k+} &= P_{vv,k-}
\end{aligned}$$

Temporal update

$$\begin{aligned}
\mathbf{x}_{\varepsilon,k+1-} &= \Phi_{\varepsilon k}\mathbf{x}_{\varepsilon,k+} \\
P_{\varepsilon\varepsilon k+1-} &= \Phi_{\varepsilon k}P_{\varepsilon\varepsilon k+}\Phi_{\varepsilon k}^{\mathrm{T}} + Q_{\varepsilon\varepsilon} \\
P_{\varepsilon v k+1-} &= \Phi_{\varepsilon k}P_{\varepsilon v k+}\Phi_{vk}^{\mathrm{T}} \\
P_{v\varepsilon k+1-} &= P_{\varepsilon v k+1-}^{\mathrm{T}} \\
P_{vvk+1-} &= \Phi_{vk}P_{vvk+}\Phi_{vk}^{\mathrm{T}} + Q_{vv}
\end{aligned}$$

8.6.4 Computational Complexity

The purpose of the Schmidt–Kalman filter is to reduce the computational requirements compared to those required for the full Kalman filter. Although the equations appear to be more complicated, the dimensions of the matrices involved are smaller than those of the matrices in the Kalman filter.

We will now do a rough operations count of those implementation equations just to be sure that they do, indeed, decrease computational requirements.

Table 8.2 is a breakdown of the operations counts for implementing the equations in Table 8.1. The formulas (in angle brackets) above the matrix formulas give the rough operations counts for implementing those formulas. An "operation" in this accounting is roughly equivalent to a multiply-and-accumulate. The operations counts are expressed in terms of the number of measurements (ℓ, the dimension of the measurement vector), the number of essential state variables (n_ε), and the number of nuisance state variables (n_v).

These complexity formulas are based on the matrix dimensions listed in Table 8.3.

TABLE 8.2 Operations Counts for the Schmidt–Kalman Filter

Scalar Operation Counts for Matrix Operations	Totals by Rows
$\mathcal{C} = \{\overbrace{H_{\varepsilon k}}^{\langle n_c\ell^2\rangle}\times(\underbrace{\overbrace{P_{\varepsilon\varepsilon k-}H_{\varepsilon k}^{\mathrm{T}}}^{\langle n_c^2\ell\rangle}+\overbrace{P_{\varepsilon v k-}H_{vk}^{\mathrm{T}}}^{\langle n_c n_v\ell\rangle}}_{\langle\text{used again below}\rangle})$	$n_\varepsilon\ell^2+n_\varepsilon^2\ell+n_\varepsilon n_v\ell$
$+\overbrace{H_{vk}}^{\langle n_v\ell^2\rangle}\times(\overbrace{P_{v\varepsilon k-}H_{\varepsilon k}^{\mathrm{T}}}^{\langle n_c n_v\ell\rangle}+\overbrace{P_{vvk-}H_{vk}^{\mathrm{T}}}^{\langle n_v^2\ell\rangle})$	$n_v\ell^2+n_\varepsilon n_v\ell+n_v^2\ell$
$+R_k\}^{-1\langle\ell^3\rangle}$(matrix inverse)	ℓ^3
$K_{SK,k}=\{\overbrace{P_{\varepsilon\varepsilon k-}H_{\varepsilon k}^{\mathrm{T}}+P_{\varepsilon v k-}H_{vk}^{\mathrm{T}}}^{\langle\text{already computed above}\rangle}\}\overbrace{\times\mathcal{C}}^{\langle n_\varepsilon\ell^2\rangle}$	$n_\varepsilon\ell^2$
$\mathrm{x}_{\varepsilon,k+}=\mathrm{x}_{\varepsilon,k-}+\overbrace{K_{\mathrm{SK},k}}^{\langle n_\varepsilon\ell\rangle}\times\overbrace{\{\mathrm{z}_k-H_{\varepsilon k}\mathrm{x}_{\varepsilon,k-}\}}^{\langle n_\varepsilon\ell\rangle}$	$2n_\varepsilon\ell$
$\mathcal{A}=l_{n_\varepsilon}-\overbrace{K_{\mathrm{SK},k}}^{\langle n_\varepsilon^2\ell\rangle}H_{\varepsilon,k}$	$n_\varepsilon^2\ell$
$\mathcal{B}=\underbrace{\overbrace{\mathcal{A}\times P_{\varepsilon v,k-}}^{\langle n_\varepsilon^2 n_v\rangle}\times\overbrace{H_{v,k}^{\mathrm{T}}\times K_{\mathrm{SK},k}^{\mathrm{T}}}^{\langle n_\varepsilon n_v\ell\rangle}}_{\langle n_\varepsilon^2 n_v\rangle}$	$2n_\varepsilon^2 n_v+n_\varepsilon n_v\ell$
$P_{\varepsilon\varepsilon,k+}=\mathcal{A}P_{\varepsilon\varepsilon,k-}\mathcal{A}^{\mathrm{T}}-\mathcal{B}-\mathcal{B}^{\mathrm{T}}$	$\frac{3}{2}n_\varepsilon^3+\frac{1}{2}n_\varepsilon^2$
$+\underbrace{K_{\mathrm{SK},k}\overbrace{[H_{vk},P_{vv,k-}H_{v,k}^{\mathrm{T}}+R_k]}^{\langle n_v^2\ell+\frac{1}{2}n_v\ell^2+\frac{1}{2}n_v\ell\rangle}K_{\mathrm{SK},k}^{\mathrm{T}}}_{\langle\frac{1}{2}n_\varepsilon^2\ell+n_\varepsilon\ell^2+\frac{1}{2}n_\varepsilon\ell\rangle}$	$\frac{3}{2}n_v^3+n_v^2\ell+\frac{1}{2}n_v\ell^2+\frac{1}{2}n_v^2+\frac{1}{2}n_v\ell$
$P_{\varepsilon v,k+}=\overbrace{\mathcal{A}\times P_{\varepsilon v,k-}}^{\langle n_\varepsilon^2 n_v\rangle}-\overbrace{K_{\mathrm{SK},k}\times H_{v,k-}}^{\langle n_\varepsilon n_v\ell\rangle}\overbrace{\times P_{vv,k-}}^{\langle n_\varepsilon n_v^2\rangle}$	$n_\varepsilon^2 n_v+n_\varepsilon n_v\ell+n_\varepsilon n_v^2$
$P_{v\varepsilon,k+}=P_{\varepsilon v,k+}^{\mathrm{T}}$	0
$P_{vv,k+}=P_{vv,k-}$	0
Total for observational update	$3n_\varepsilon\ell^2+\frac{5}{2}n_\varepsilon^2\ell+4n_\varepsilon n_v\ell$ $+\frac{3}{2}n_v\ell^2+2n_v^2\ell+\ell^3$ $+\frac{5}{2}n_\varepsilon\ell+3n_\varepsilon^2 n_v$ $+\frac{1}{2}n_v\ell+n_\varepsilon n_v^2$

(*Continued*)

TABLE 8.2 *Continued*

Scalar Operation Counts for Matrix Operations	Totals by Rows
Temporal update	
$\hat{\mathbf{x}}_{\varepsilon,k+1-} = \Phi_{\varepsilon k}\hat{\mathbf{x}}_{\varepsilon,k+}$	n_ε^2
$P_{\varepsilon\varepsilon k+1-} = \Phi_{\varepsilon k}P_{\varepsilon\varepsilon k+}\Phi_{\varepsilon k}^{\mathrm{T}} + Q_{\varepsilon\varepsilon}$	$\frac{3}{2}n_\varepsilon^3 + \frac{1}{2}n_\varepsilon^2$
$P_{\varepsilon v k+1-} = \Phi_{\varepsilon k}P_{\varepsilon v k+}\Phi_{vk}^{\mathrm{T}}$	$n_\varepsilon n_v^2 + n_v n_\varepsilon^2$
$P_{v\varepsilon k+1-} = P_{\varepsilon v k+1-}^{\mathrm{T}}$	0
$P_{vvk+1-} = \Phi_{vk}P_{vvk+}\Phi_{vk}^{\mathrm{T}} + Q_{vv}$	$\frac{3}{2}n_v^3 + \frac{1}{2}n_v^2$
Total for temporal update	$\frac{3}{2}n_\varepsilon^2 + \frac{3}{2}n_\varepsilon^3 + n_\varepsilon n_v^2 + n_\varepsilon^2 n_v + \frac{3}{2}n_v^3 + \frac{1}{2}n_v^2$
Total for Schmidt–Kalman filter	$3n_\varepsilon\ell^2 + \frac{5}{2}n_\varepsilon^2\ell + 4n_\varepsilon n_v\ell$ $+\frac{3}{2}n_v\ell^2 + 2n_v^2\ell + \ell^3 + \frac{5}{2}n_\varepsilon\ell$ $+4n_\varepsilon^2 n_v + \frac{3}{2}n_\varepsilon^3 + \frac{3}{2}n_\varepsilon^2 + \frac{1}{2}n_v\ell$ $+\frac{3}{2}n_v^3 + \frac{1}{2}n_v^2 + \frac{1}{2}n_v\ell + 2n_\varepsilon n_v^2$

TABLE 8.3 Array Dimensions

Symbol	Rows	Columns
$\mathcal{A}$	n_ε	n_ε
$\mathcal{B}$	n_ε	n_ε
$\mathcal{C}$	ℓ	ℓ
H_ε	ℓ	n_ε
H_v	ℓ	n_v
K_{SK}	n_ε	ℓ
$P_{\varepsilon\varepsilon}$	n_ε	n_ε
$P_{\varepsilon v}$	n_ε	n_v
$P_{v\varepsilon}$	n_v	n_ε
P_{vv}	n_v	n_v
$Q_{\varepsilon\varepsilon}$	n_ε	n_ε
Q_{vv}	n_v	n_v
R	ℓ	ℓ
Φ_ε	n_ε	n_ε
Φ_v	n_v	n_v
$\hat{x}_\varepsilon$	n_ε	1
z	ℓ	1

A MATLAB implementation of the Schmidt-Kalman filter is in the m-file `KFvsSKF.m` on the accompanying CD.

8.7 MEMORY, THROUGHPUT, AND WORDLENGTH REQUIREMENTS

These may not be important issues for off-line implementations of Kalman filters on mainframe scientific computers, but they can become critical issues for real-time implementations in embedded processors, especially as the dimensions of the state vector or measurement become larger. We present here some methods for assessing these requirements for a given application and for improving feasibility in marginal cases. These include order-of-magnitude plots of memory requirements and computational complexity as functions of the dimensions of the state vector and measurement vector. These plots cover the ranges from 1 to 1000 for these dimensions, which should include most problems of interest.

8.7.1 Wordlength Problems

Precision Problems Wordlength issues include precision problems (related to the number of significant bits in the mantissa field) and dynamic range problems (related to the number of bits in the exponent field). The issues and remedies related to precision are addressed in Chapter 6.

Scaling Problems Underflows and overflows are symptoms of dynamic range problems. These can often be corrected by rescaling the variables involved. This is equivalent to changing the units of measure, such as using kilometers in place of centimeters to represent length. In some but not all cases, the condition number of a matrix can be improved by rescaling. For example, the two covariance matrices

$$\begin{bmatrix} 1 & 0 \\ 0 & \varepsilon^2 \end{bmatrix} \text{ and } \frac{1}{2}\begin{bmatrix} 1+\varepsilon^2 & 1-\varepsilon^2 \\ 1-\varepsilon^2 & 1+\varepsilon^2 \end{bmatrix}$$

have the same condition number $(1/\varepsilon^2)$, which can be troublesome for very small values of ε. The condition number of the matrix on the left can be made equal to 1 by simply rescaling the second component by $1/\varepsilon$.

8.7.2 Memory Requirements

In the early years of Kalman filter implementation, a byte of memory cost about as much as a labor hour at minimum wage. With these economic constraints, programmers developed many techniques for reducing the memory requirements of Kalman filters. A few of these techniques have been mentioned in Chapter 6, although they are not as important as they once were. Memory costs have dropped dramatically since these methods were developed. The principal reason for paying attention to

memory requirements nowadays is to determine the limits on problem size with a fixed memory allocation. Memory is cheap but still finite.

Program Memory versus Data Memory In the *von Neumann architecture* for processing systems, there is no distinction between the memory containing the algorithms and that containing the data used by the algorithms. In specific applications, the program may include formulas for calculating the elements of arrays such as Φ or H. Other than that, the memory requirements for the algorithms tend to be independent of the application and the problem size. For the Kalman filter, the problem size is specified by the dimensions of the state (n), measurement (ℓ), and process noise (p). The data memory requirements for storing the arrays with these dimensions are very much dependent on the problem size. We present here some general formulas for this dependence.

Data Memory and Wordlength The data memory requirements will depend upon the data wordlengths (in bits) as well as the size of the data structures. The data requirements are quoted in *floating-point words*. These are either 4- or 8-byte words (in IEEE floating-point standard formats) for the examples presented in this book.

Data Memory Requirements These are also influenced somewhat by programming style, particularly by the ways that data structures containing partial results are reused.

The data memory requirements for a more-or-less conventional implementation of the Kalman filter are listed in Table 8.4 and plotted in Figure 8.25. This is the Kalman filter implementation diagrammed in Figure 6.2, which reuses some partial results.

TABLE 8.4 Array Requirements for Conventional Kalman Filter Implementation

Functional Grouping	Matrix Expression	Array Dimensions	Total Memory*
Riccati equation	P	$n \times n$	
	ΦP	$n \times n$	
	GQG^{T}	$n \times n$	$3n^2$
	HP PH^{T}	$\ell \times n$	$+\ell n$
	R	$\ell \times \ell$	
	$HPH^{\mathrm{T}} + R$ $[HPH^{\mathrm{T}} + R]^{-1}$	$\ell \times \ell$	$+2\ell^2$
Common	Φ	$n \times n$	n^2
	H	$\ell \times n$	
	$\overline{K}$	$n \times \ell$	$+2\ell n$
Linear estimation	z $z - Hx$	ℓ	ℓ
	x	n	
	Φ_x	n	$+2n$

*In units of floating-point data words.

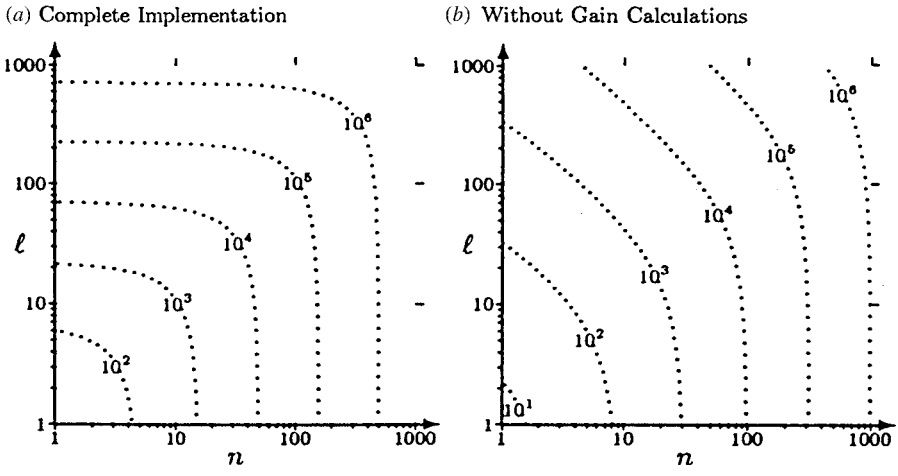


Figure 8.25 Conventional filter memory requirements (in words) versus state dimension (n) and measurement dimension (ℓ).

The array dimensions are associated with the results of matrix subexpressions that they contain. These are divided into three groups:

1. those arrays common to both the Riccati equation (for covariance and gain computations) and the linear estimation computations;
2. the additional array expressions required for solving the Riccati equation, which provides the Kalman gain as a partial result; and
3. the additional array expressions required for linear estimation of the state variables, given the Kalman gains.

Expressions grouped together by curly braces are assumed to be kept in the same data structures. This implementation assumes that

- the product GQG^{T} is input and not computed (which eliminates any dependence on p, the dimension of Q);
- given ΦP, Φ, and GQG^{T}, the operation $P \leftarrow \Phi P \Phi^{\mathrm{T}} + GQG^{\mathrm{T}}$ is performed in place;
- computations involving the subexpression PH^{T} can be implemented with HP by changing the indexing;
- $HPH^{\mathrm{T}} + R$ can be computed in place (from HP, H, and R) and inverted in place;
- $z - Hx$ can be computed in place (in the z array); and
- the state update computation $x \leftarrow (\Phi x) + \overline{K}[z - H(\Phi x)]$ requires additional memory only for the intermediate result (Φx).

Figure 8.25 illustrates the numerical advantage of precomputing and storing the Kalman gains in bulk memory. It saves about a factor of 4 in data memory

requirements for small dimensions of the measurement relative to the state and even more for larger measurement dimensions.

Eliminating Data Redundancy Reuse of temporary arrays is not the only way to save memory requirements. It is also possible to save data memory by eliminating redundancies in data structures. The symmetry of covariance matrices is an example of this type of redundancy. The methods discussed here depend on such constraints on matrices that can be exploited in designing their data structures. They do require additional programming effort, however, and the resulting runtime program code may require slightly more memory and more processing time. The difference will involve primarily index computations, which are not the standard ones used by optimizing compilers. Table 8.5 lists some common constraints on square matrices, the minimum memory requirements (as multiples of the memory required for a scalar variable), and corresponding indexing schemes for packing the matrices in singly subscripted arrays. The indexing schemes are given as formulas for the single subscript (k) corresponding to the row (i) and column (j) indices of a two-dimensional array. The two formulas given for symmetric matrices correspond to the two alternative indexing schemes

$$\begin{bmatrix} 1 & 2 & 4 & \dots & \frac{1}{2}n(n-1)+1 \\ & 3 & 5 & \dots & \frac{1}{2}n(n-1)+2 \\ & & 6 & \dots & \frac{1}{2}n(n-1)+3 \\ & & & \ddots & \vdots \\ & & & \dots & \frac{1}{2}n(n+1) \end{bmatrix} \text{ or } \begin{bmatrix} 1 & 2 & 3 & \dots & n \\ & n+1 & n+2 & \dots & 2n-1 \\ & & 2n & \dots & 3n-3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ & & & \dots \frac{1}{2}n(n+1) & \end{bmatrix},$$

where the element in the ith row and jth column is $k(i, j)$.

TABLE 8.5 Minimum Memory Requirements for $n \times n$ Matrices*

Matrix Type	Minimum Memory†	Indexing $k(i, j)$
Symmetric	$\frac{n(n+1)}{2}$	$i + \frac{j(j-1)}{2}$
		$\frac{(2n-i)(i-1)}{2} + j$
Upper triangular	$\frac{n(n+1)}{2}$	$i + \frac{j(j-1)}{2}$
Unit upper triangular	$\frac{n(n+1)}{2}$	$i + \frac{(j-1)(j-2)}{2}$
Strictly upper triangular	$\frac{n(n-1)}{2}$	$i + \frac{(j-1)(j-2)}{2}$
Diagonal	n	i
Toeplitz	n	$i + j - 1$

Note: n is the dimension of the matrix; i and j are the indices of a two-dimensional array; k is the corresponding index of a one-dimensional array.
†In units of data words.

Just by exploiting symmetry or triangularity, these methods can save about a factor of 2 in memory with a fixed state vector dimension or allow about a factor of $\sqrt{2}$ increase in the state vector dimension (about a 40% increase in the dimension) with the same amount of memory.

Arrays Can Be Replaced by Algorithms In special cases, data arrays can be eliminated entirely by using an algorithm to compute the matrix elements "on the fly." For example, the companion form coefficient matrix (F) and the corresponding state transition matrix (Φ) for the differential operator d^n/dt^n are

$$F = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}, \tag{8.142}$$

$$\Phi(t) = e^{Ft} \tag{8.143}$$

$$= \begin{bmatrix} 1 & t & \frac{1}{2}t^2 & \cdots & \frac{1}{(n-1)!}t^{n-1} \\ 0 & 1 & t & \cdots & \frac{1}{(n-2)!}t^{n-2} \\ 0 & 0 & 1 & \cdots & \frac{1}{(n-3)!}t^{n-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}, \tag{8.144}$$

where t is the discrete-time interval. The following algorithm computes the product $M = \Phi P$, with Φ as given in Equation 8.144, using only P and t:

```
for i = 1: n,
  for j = 1: n,
      s = P (n, j);
      m = n - 1;
      for k = n - 1: -1: i,
          s = P (k, j) + s * t/m;
          m = m - 1;
      end;
      M (i, j) = s;
  end;
end;
```

It requires about half as many arithmetic operations as a general matrix multiply and requires no memory allocation for one of its matrix factors.

8.7.3 Throughput, Processor Speed, and Computational Complexity

Influence of Computational Complexity on Throughput The throughput of a Kalman filter implementation is related to how many updates it can perform in a unit of time. This depends upon the speed of the host processor, in floating-point operations (flops) per second, and the computational complexity of the application, in flops per filter update:

$$\text{throughput}\left(\frac{\text{updates}}{\text{s}}\right) = \frac{\text{processor speed (flops/s)}}{\text{computational complexity (flops/update)}}.$$

The numerator of the expression on the right-hand side depends upon the host processor. Formulas for the denominator, as functions of the application problem size, are derived in Chapter 6. These are the *maximum* computational complexities of the implementation methods listed in Chapter 6. The computational complexity of an application can be made smaller if it includes sparse matrix operations that can be implemented more efficiently.

Conventional Kalman Filter The maximum computational complexity of the conventional Kalman filter implementation is plotted versus problem size in Figure 8.26. This implementation uses all the shortcuts listed in Section 8.7.2 and eliminates redundant computations in symmetric matrix products. The right-hand plot assumes that the matrix Riccati equation for the Kalman gain computations has been solved off-line, as in a gain-scheduled or steady-state implementation.

Bierman–Thornton Square Root Implementation The corresponding dependence of computational complexity on problem size for the *UD* filter implementation is

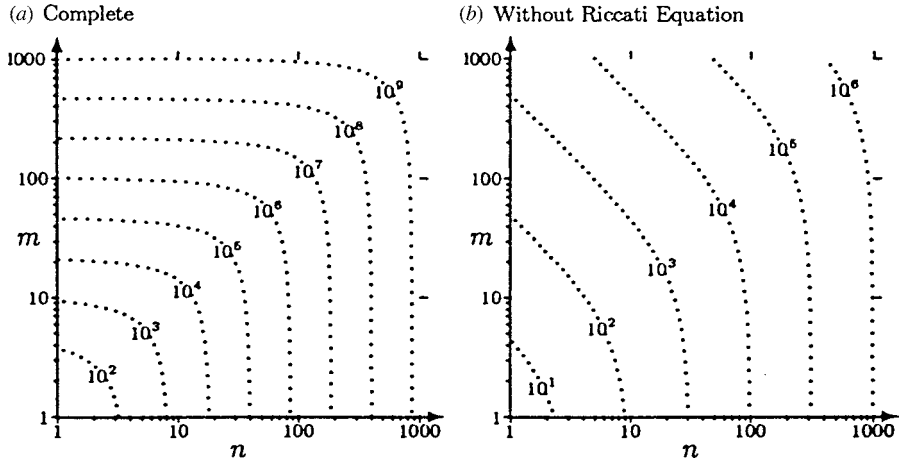


Figure 8.26 Contour plots of computational complexity (in flops per measurement) of the Kalman filter as a function of state dimension (n) and measurement dimension (m).

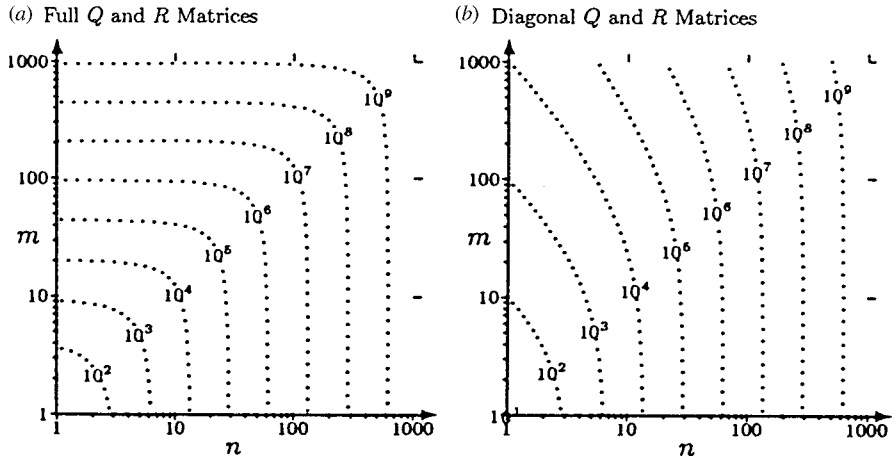


Figure 8.27 Contour plots of computational complexity (in flops per measurement) of the Bierman–Thornton implementation as a function of state dimension (n) and measurement dimension (m).

plotted in Figure 8.27(a). These data include the computational cost of diagonalizing Q and R on each temporal and observational update, respectively. The corresponding results for the case where Q and R are already diagonal are displayed in Figure 8.27(b).

8.7.4 Programming Cost versus Runtime Cost

The computational complexity issue in Kalman filtering is usually driven by the need to execute in real time. Computational complexity grows so fast with problem size that it will overwhelm even the fastest processors for sufficiently large system models. For that reason, computational complexity is an issue that must be addressed early on in the filter design cycle.

Another trade-off in the design of Kalman filters is between the one-time cost of programming the implementation and the recurring cost of running it on a computer. As computers grow less expensive compared to programmers, this trade-off tends to favor the most straightforward methods, even those that cause numerical analysts to wince. Keep in mind, however, that this is a low-cost/high-risk approach. Remember that the reason for the development of better implementation methods was the failure of straightforward programming solutions to produce acceptable results.

8.8 WAYS TO REDUCE COMPUTATIONAL REQUIREMENTS

8.8.1 Reducing Complexities of Matrix Products

Implementing Products of Two Matrices The number of flops required to compute the product of general $\ell \times m$ and $m \times n$ matrices is ℓmn. This figure can be reduced

TABLE 8.6 Computational Complexities of a Triple Matrix Product

Attribute	Value	
Implementation	$\underbrace{M_1}_{n_1\times n_2} \times (\underbrace{M_2}_{n_2\times n_3} \times \underbrace{M_3}_{n_3\times n_4})$	$(\underbrace{M_1}_{n_1\times n_2} \times \underbrace{M_2}_{n_2\times n_3}) \times \underbrace{M_3}_{n_3\times n_4}$
Number of flops: First, multiply	$n_2n_3n_4$	$n_1n_2n_3$
Second, multiply	$n_1n_2n_4$	$n_1n_3n_4$
Total	$n_2(n_3 + n_1)n_4$	$n_1(n_2 + n_4)n_3$

substantially for matrices with predictable patterns of zeros or symmetry properties. These tricks can be used to advantage in computing matrix products that are symmetric and products involving diagonal or triangular factors. They should always be exploited whenever H or Φ is a sparse matrix.

Implementing Products of Three Matrices It is of considerable practical importance that *associativity of matrix multiplication does not imply invariance of computational complexity*. The associativity of matrix multiplication is the property that

$$M_1 \times (M_2 \times M_3) = (M_1 \times M_2) \times M_3 \tag{8.145}$$

for conformably dimensioned matrices M_1, M_2, and M_3. That is, the *result* is guaranteed to be independent of the *order* in which the two matrix multiplications are performed. However, the *effort* required to obtain the result is *not* always independent of the order of multiplication. This distinction is evident if one assigns conformable dimensions to the matrices involved and evaluates the number of scalar multiplications required to compute the result, as shown in Table 8.6. The number of flops depends on the order of multiplication, being $n_2(n_3 + n_1)n_4$ in one case and $n_1(n_2 + n_4)n_3$ in the other case. The implementation $M_1 \times (M_2 \times M_3)$ is favored if $n_1n_2(n_4 - n_3) < (n_1 - n_2)n_3n_4$, and the implementation $(M_1 \times M_2) \times M_3$ is favored if the inequality is reversed. The correct selection is used to advantage in the more practical implementations of the Kalman filter, such as the De Vries implementation (see Section 6.6.1.4).

8.8.2 Off-Line versus On-Line Computational Requirements

The Kalman filter is a real-time algorithm in the sense that it calculates an estimate of the current state of a system given measurements obtained in real time. In order that the filter be implementable in real time, however, it must be possible to execute the algorithm in real time with the available computational resources. In this assessment, it is important to distinguish between those parts of the filter algorithm that must be performed on-line and those that can be performed off-line (i.e., carried out beforehand, with the results stored in memory, including bulk media, such as

magnetic tape or a CD-ROM, and read back in real time[6]). The on-line computations are those that depend upon the measurements of the real-time system. Those calculations cannot be made until their input data become available.

It is noted in Chapter 4 that the computations required for calculating the Kalman gains do not depend upon the real-time data, and for that reason they can be executed off-line. This is repeated here for emphasis and to formalize some of the practical methods used for implementation.

The most straightforward method is to precompute the gains and store them for retrieval in real time. This is also the method with the most general applicability. Some methods with greater efficiency (but less generality) are discussed in the following subsections. Methods for performance analysis of these suboptimal estimation methods are discussed in Section 8.5.

8.8.3 Gain Scheduling

This is an approximation method for estimation problems in which the rate of change of the Kalman gains is very slow compared to the sampling rate. Typically, the relative change in the Kalman gain between observation times may be a few percent or less. In that case, one value of the Kalman gain may be used for several observation times. Each gain value is used for a stage of the filtering process.

This approach is typically used for problems with constant coefficients. The gains in this case have an asymptotic constant value but go through an initial transient phase due to larger or smaller initial uncertainties than the steady-state uncertainties. A few "staged" values of the gains during that transient phase may be sufficient to achieve adequate performance. The values used may be sampled values within the stage in which they are used or weighted averages of all the exact values over the range.

The performance trade-off between decreased storage requirements (for using fewer values of the gains) and increased approximation error (due to differences between optimal gains and scheduled gains) can be analyzed by simulation.

8.8.4 Steady-State Gains for Time-Invariant Systems

This is the limiting case of gain scheduling—with only one stage—and it is one of the more common uses of the algebraic Riccati equation. In this case, only the asymptotic values of the gains are used. This requires the solution of the algebraic (steady-state) matrix Riccati equation.

Several methods for solving the steady-state matrix Riccati equation are presented in the following subsections. One of these (the doubling method) is based on the linearization method for the Riccati equation presented in Chapter 4. Theoretically, it converges exponentially faster than the serial iteration method. In practice, however, convergence can stall (due to numerical problems) before an accurate

[6]In assessing the real-time implementation requirements, one must trade off the time required to read these prestored values versus the time required to compute them. In some cases, the read times may exceed the computation times.

solution is attained. However, it can still be used to obtain a good starting estimate for the Newton–Raphson method (described in Chapter 4).

8.8.4.1 Doubling Method for Time-Invariant Systems This is an iterative method for approximating the asymptotic solution to the *time-invariant* Riccati equation based on the formula given in Lemma 2 in Chapter 4. As in the continuous case, the asymptotic solution should equal the solution of the *steady-state equation*:

$$P_\infty = \Phi[P_\infty - P_\infty H^{\mathrm{T}}(HP_\infty H^{\mathrm{T}} + R)^{-1} HP_\infty]\Phi^{\mathrm{T}} + Q, \tag{8.146}$$

although this is not the form of the equation that is used. Doubling methods generate the sequence of solutions

$$P_1(-), P_2(-), P_4(-), P_8(-), \ldots, P_{2^k}(-), P_{2^{k+1}}(-), \ldots$$

of the nonalgebraic matrix Riccati equation as an initial-value problem—by doubling the time interval between successive solutions. The doubling speedup is achieved by successive squaring of the equivalent state transition matrix for the time-invariant Hamiltonian matrix

$$\Psi = \begin{bmatrix} (\Phi + Q\Phi^{-\mathrm{T}} HR^{-1}H^{\mathrm{T}}) & Q\Phi^{-\mathrm{T}} \\ \Phi^{-\mathrm{T}} H^{\mathrm{T}} R^{-1} H & \Phi^{-\mathrm{T}} \end{bmatrix}. \tag{8.147}$$

The pth squaring of Ψ will then yield Ψ^{2p} and the solution

$$P_{2p}(-) = A_{2p} B_{2p}^{-1} \tag{8.148}$$

for

$$\begin{bmatrix} A_{2p} \\ B_{2p} \end{bmatrix} = \Psi^{2p} \begin{bmatrix} A_0 \\ B_0 \end{bmatrix}. \tag{8.149}$$

Davison–Maki–Friedlander–Kailath Squaring Algorithm Note that if one expresses Ψ^{2^N} in symbolic form as

$$\Psi^{2^N} = \begin{bmatrix} \mathcal{A}_N^{\mathrm{T}} + \mathcal{C}_N \mathcal{A}_N^{-1} \mathcal{B}_N & \mathcal{C}_N \mathcal{A}_N^{-1} \\ \mathcal{A}_N^{-1} \mathcal{B}_N & \mathcal{A}_N^{-1} \end{bmatrix}, \tag{8.150}$$

then its square can be put in the form

$$\Psi^{2^{N+1}} = \begin{bmatrix} \mathcal{A}_N^{\mathrm{T}} + \mathcal{C}_N \mathcal{A}_N^{-1} \mathcal{B}_N & \mathcal{C}_N \mathcal{A}_N^{-1} \\ \mathcal{A}_N^{-1} \mathcal{B}_N & \mathcal{A}_N^{-1} \end{bmatrix}^2 \tag{8.151}$$

$$= \begin{bmatrix} \mathcal{A}_{N+1}^{\mathrm{T}} + \mathcal{C}_{N+1} \mathcal{A}_{N+1}^{-1} \mathcal{B}_{N+1} & \mathcal{C}_{N+1} \mathcal{A}_{N+1}^{-1} \\ \mathcal{A}_{N+1}^{-1} \mathcal{B}_{N+1} & \mathcal{A}_{N+1}^{-1} \end{bmatrix}, \tag{8.152}$$

$$\mathcal{A}_{N+1} = \mathcal{A}_N (I + \mathcal{B}_N \mathcal{C}_N)^{-1} \mathcal{A}_N^{\mathrm{T}}, \tag{8.153}$$

$$\mathcal{B}_{N+1} = \mathcal{B}_N + \mathcal{A}_N (I + \mathcal{B}_N \mathcal{C}_N)^{-1} \mathcal{B}_N \mathcal{A}_N^{\mathrm{T}}, \tag{8.154}$$

$$\mathcal{C}_{N+1} = \mathcal{C}_N + \mathcal{A}_N^{\mathrm{T}} \mathcal{C}_N (I + \mathcal{B}_N \mathcal{C}_N)^{-1} \mathcal{A}_N. \tag{8.155}$$

The last three equations define an algorithm for squaring Ψ^{2^N}, starting with the values of $\mathcal{A}_N$, $\mathcal{B}_N$, and $\mathcal{C}_N$ for $N = 0$, given by Equation 8.147:

$$\mathcal{A}_0 = \Phi^{\mathrm{T}}, \tag{8.156}$$

$$\mathcal{B}_0 = H^{\mathrm{T}} R^{-1} H, \tag{8.157}$$

$$\mathcal{C}_0 = Q. \tag{8.158}$$

Initial Conditions If the initial value of the Riccati equation is $P_0 = 0$, the zero matrix, it can be represented by $P_0 = A_0 B_0^{-1}$ for $A_0 = 0$ and any nonsingular B_0. Then the Nth iterate of the doubling algorithm will yield

$$\begin{bmatrix} A_{2^N} \\ B_{2^N} \end{bmatrix} = \Psi^{2^N} \begin{bmatrix} A_0 \\ B_0 \end{bmatrix} \tag{8.159}$$

$$= \begin{bmatrix} \mathcal{A}_N^{\mathrm{T}} + \mathcal{C}_N \mathcal{A}_N^{-1} \mathcal{B}_N & \mathcal{C}_N \mathcal{A}_N^{-1} \\ \mathcal{A}_N^{-1} \mathcal{B}_N & \mathcal{A}_N^{-1} \end{bmatrix} \begin{bmatrix} 0 \\ B_1 \end{bmatrix} \tag{8.160}$$

$$= \begin{bmatrix} \mathcal{C}_N \mathcal{A}_N^{-1} \mathcal{B}_1 \\ \mathcal{A}_N^{-1} B_1 \end{bmatrix}, \tag{8.161}$$

$$P_{2^N} = A_{2^N} B_{2^N}^{-1} \tag{8.162}$$

$$= \mathcal{C}_N \mathcal{A}_N^{-1} B_1 (\mathcal{A}_N^{-1} B_1)^{-1} \tag{8.163}$$

$$= \mathcal{C}_N. \tag{8.164}$$

That is, after the Nth squaring step, the submatrix

$$\mathcal{C}_N = P_2^N. \tag{8.165}$$

The resulting algorithm is summarized in Table 8.7. It has complexity $\mathcal{O}(n^3 \log k)$ flops for computing P_k, requiring one $n \times n$ matrix inverse and eight $n \times n$ matrix products per iteration.[7] An array allocation scheme for performing the squaring algorithm using only six $n \times n$ arrays is shown in Figure 8.28.

Numerical Convergence Problems Convergence can be stalled by precision limitations before it is complete. The problem is that the matrix $\mathcal{A}$ is effectively squared on each iteration and appears quadratically in the update equations for $\mathcal{B}$ and $\mathcal{C}$. Consequently, if $\|\mathcal{A}_N\| \ll 1$, then the computed values of $\mathcal{B}_N$ and $\mathcal{C}_N$ may become stalled numerically as $\|\mathcal{A}_N\| \to 0$ exponentially. The value of $\mathcal{A}_N$ can be monitored to test for this stall condition. Even in stall situations, the doubling algorithm is still an efficient method for getting an approximate nonnegative definite solution.

[7]The matrices $\mathcal{B}_N$ and $\mathcal{C}_N$ and the matrix products $(I + \mathcal{B}_N \mathcal{C}_N)^{-1} \mathcal{B}_N$ and $\mathcal{C}_N (I + \mathcal{B}_N \mathcal{C}_N)^{-1}$ are symmetric. That fact can be exploited to eliminate $2n^2(n-1)$ flops per iteration. With these savings, this algorithm requires slightly fewer flops per iteration than the straightforward squaring method. It also requires about one-fourth less memory than straightforward squaring.

TABLE 8.7 Davison–Maki–Friedlander–Kailath Squaring Algorithm

Initialization	Iteration (N times)
$\mathcal{A} = \Phi^{\mathrm{T}}$	$\mathcal{A} \leftarrow \mathcal{A}(I + \mathcal{BC})^{-1}\mathcal{A}^{\mathrm{T}}$
$\mathcal{B} = H^{\mathrm{T}}R^{-1}H$	$\mathcal{B} \leftarrow \mathcal{B} + \mathcal{A}(I + \mathcal{BC})^{-1}\mathcal{BA}^{\mathrm{T}}$
$\mathcal{C} = Q(= P_1)$	$\mathcal{C} \leftarrow \mathcal{C} + \mathcal{A}^{\mathrm{T}}\mathcal{C}(I + \mathcal{BC})^{-1}\mathcal{A}$
Termination	
$P_{2^N} = C$	

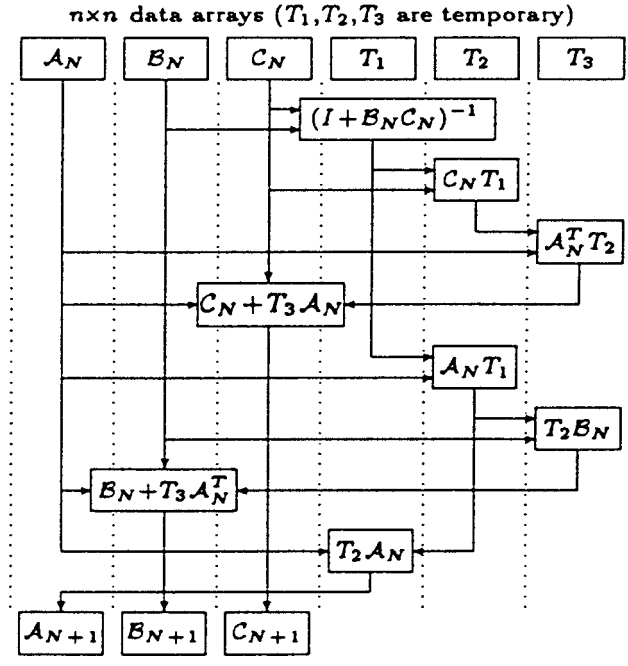


Figure 8.28 Data array usage for the doubling algorithm.

8.9 ERROR BUDGETS AND SENSITIVITY ANALYSIS

8.9.1 Design Problem for Statistical Performance Requirements

This is the problem of estimating the statistical performance of a sensor system that will make measurements of some dynamic and stochastic process and estimate its state. Statistical performance is defined by mean-squared errors at the system level; these depend on mean-squared errors at the subsystem level, and so on, down to the level of individual sensors and components. The objective of this

activity is to be able to justify the apportionment of these lower-level performance requirements.

This type of performance analysis is typically performed during the preliminary design of estimation systems. The objective of the analysis is to evaluate the feasibility of an estimation system design for meeting some prespecified acceptable level of uncertainty in the estimates that will be obtained.

The Kalman filter does not design sensor systems, but it provides the tool for doing it defensibly. That tool is the model for estimation uncertainty. The covariance propagation equations derived from the model can be used in characterizing estimation uncertainty as a function of the parameters of the design. Some of these parameters are statistical, such as the noise models of the sensors under consideration. Others are deterministic. The deterministic parameters may also be discrete valued—such as the sensor type—or continuous—such as the sensor location.

One of the major uses of Kalman filtering theory is in the design of sensor systems:

1. Vehicle navigation systems containing some combination of sensors, such as:
   (a) Attitude and attitude rate sensors
      i. Magnetic compass (field sensor)
      ii. Displacement gyroscopes
      iii. Star trackers or sextants
      iv. Rate gyroscopes
      v. Electric field sensors (for earth potential field)
   (b) Acceleration sensors (accelerometers)
   (c) Velocity sensors (e.g., onboard Doppler radar)
   (d) Position sensors
      i. Global navigation satellite system (GNSS)
      ii. Terrain-mapping radar
      iii. Long-range navigation (LORAN)
      iv. Instrument landing system (ILS)
2. Surface-based, airborne, or spaceborne tracking systems
   (a) Range and Doppler radar
   (b) Imaging sensors (e.g., visible or infrared cameras)

In the design of these systems, it is assumed that a Kalman filter will be used in estimating the dynamic state (position and velocity) of the vehicle. Therefore, the associated covariance equations can be used to estimate the performance in terms of the covariance of estimation uncertainty.

8.9.2 Error Budgeting

Large systems such as spacecraft and aircraft contain many sensors of many types, and the Kalman filter provides a methodology for the integrated design of such

systems. Error budgeting is a specialized form of sensitivity analysis. It uses the error covariance equations of the Kalman filter to formalize the dependence of system accuracy on the component accuracies of its individual sensors. This form of covariance analysis is significantly more efficient than Monte Carlo analysis for this purpose, although it does depend upon linearity of the underlying dynamic processes.

Error budgeting is a process for trading off performance requirements among sensors and subsystems of a larger system for meeting a diverse set of overall performance constraints imposed at the system level. The discussion here is limited to the system-level requirements related to *accuracy*, although most system requirements include other factors related to cost, weight, size, and power.

The error budget is an allocation of accuracy requirements down through the hierarchy of subsystems to individual sensors and even to their component parts. It is used for a number of purposes, such as:

1. Assessing theoretical or technological performance limits by determining whether the performance requirements for a given application of a given system are *achievable* within the performance capabilities of available, planned, or theoretically attainable sensor subsystems.
2. Determining the extent of *feasible design space*, which is the range of possible sensor types and their design parameters (e.g., placement, orientation, sensitivity, and accuracy) for meeting the system performance requirements.
3. Finding a *feasible apportionment* of individual subsystem or sensor accuracies for meeting overall system accuracy requirements.
4. Identifying the *critical* subsystems, that is, those for which slight degradation or derating of performance would most severely affect system performance. These are sometimes called the *long poles in the tent*, because they tend to "stick out" in this type of assessment.
5. Finding feasible *upgrades and redesigns* of existing systems for meeting new performance requirements. This may include relaxation of some requirements and tightening of others.
6. *Trading off* requirements among subsystems. This is done for a number of reasons:
   (a) Reapportionment of error budgets to meet a new set of requirements (item 5 above).
   (b) Relaxing accuracy requirements where they are difficult (or expensive) to attain and compensating by tightening requirements where they are easier to attain. This approach can sometimes be used to overcome sensor problems uncovered in concurrent sensor development and testing.
   (c) Reducing other system-level performance attributes, such as cost, size, weight, and power. This also includes such practices as using suboptimal filtering methods to reduce computational requirements.

8.9.3 Error Budget

Multiple Performance Requirements System-level performance requirements can include constraints on the mean-squared values of several error types at several different times. For example, the navigational errors of a space-based imaging system may be constrained at several points corresponding to photographic missions or planetary encounters. These constraints may include errors in pointing (attitude), position, and velocity. The error budget must then consider how each component, component group, or subsystem contributes to each of these performance requirements. The budget will then have a two-dimensional breakout—like a spreadsheet—as shown in Figure 8.29. The rows represent the contributions of major sensor subsystems, and the columns represent their contributions to each of the multiple system-level error constraints. The formulas determining how each error source contributes to each of the system-level error categories are more complex than those of the usual spreadsheet, however.

8.9.4 Error Sensitivity Analysis and Budgeting

Nonlinear Programming Problem The dependence of mean-squared system-level errors on mean-squared subsystem-level errors is nonlinear, and the budgeting process seeks a satisfactory apportionment of the subsystem-level error covariances by a gradient-like method. This includes sensitivity analysis to determine the gradients of the various mean-squared system-level errors with respect to the mean-squared subsystem-level errors.

Dual-State System Model Errors considered in the error budgeting process may include known modeling errors due to simplifying assumptions or other measures to reduce the computational burden of the filter. For determining the effects that

ERROR BUDGET					
ERROR SOURCE GROUP	SYSTEM ERRORS				
	E_1	E_2	E_3	...	E_n
G_1				...	
G_2				...	
G_3				...	
⋮	⋮	⋮	⋮	⋮	⋮
G_m				...	
TOTAL				...	

Figure 8.29 Error budget breakdown.

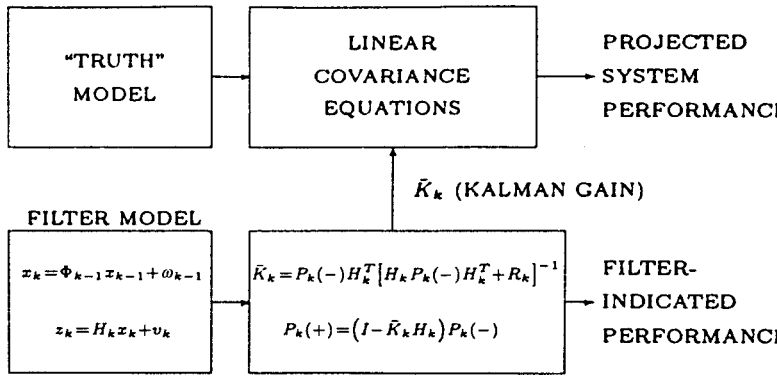


Figure 8.30 Error budget model.

errors of this type will have on system performance, it is necessary to carry both models in the analysis: the *truth model* and the *filter model.* The budgeting model used in this analysis is diagrammed in Figure 8.30. In sensitivity analysis, equivalent variations of some parameters must be made in both models. The resulting variations in the projected performance characteristics of the system are then used to establish the sensitivities to the corresponding variations in the subsystems. These sensitivities are then used to plan how to modify the current *protobudget* to arrive at an error budget allocation that will meet all performance requirements. Often, this operation must be repeated many times, because the sensitivities estimated from variations are accurate only for small changes in the budget entries.

There Are Two Stages of the Budgeting Process The first stage results in a *sufficing* error budget. It should meet system-level performance requirements and be reasonably close to attainable subsystem-level performance capabilities. The second stage includes *finessing* these subsystem-level error allocations to arrive at a more reasonable distribution.

8.9.5 Budget Validation by Monte Carlo Analysis

It is possible to validate some of the assumptions used in the error budgeting process by analytical and empirical methods. Although covariance analysis is more efficient for developing the error budget, Monte Carlo analysis is useful for assessing the effects of nonlinearities that have been approximated by variational models. This is typically done after the error budget is deemed satisfactory by linear methods. Monte Carlo analysis can then be performed on a dispersion of actual trajectories about some nominal trajectory to test the validity of the results estimated from the nominal trajectory. This is the only way to test the influence of nonlinearities, but it can be computationally expensive. Typically, many Monte Carlo runs must be made to obtain reasonable confidence in the results.

Monte Carlo analysis has certain advantages over covariance analysis, however. Monte Carlo simulations can be integrated with actual hardware, for example, to test the system performance in various stages of development. This is especially useful for testing filter performance in onboard computer implementations using actual system hardware as it becomes available. Sign errors in the filter algorithms that may be unimportant in covariance analysis tend to show up under these test conditions.

8.10 OPTIMIZING MEASUREMENT SELECTION POLICIES

8.10.1 Measurement Selection Problem

Relation to Kalman Filtering and Error Budgeting You have seen how Kalman filtering solves the optimization problem related to the *use* of data obtained from a measurement and how error budgeting is used to quantify the relative merits of alternative sensor designs. However, there is an even more fundamental optimization problem related to the *selection* of those measurements. This is not an estimation problem, strictly speaking, but a *decision problem*. It is usually considered to be a problem in the general theory of optimal control, because the decision to make a measurement is considered to be a generalized control action. The problem can also be ill-posed, in the sense that there may be no unique optimal solution [12].

Optimization with Respect to a Quadratic Loss Function The Kalman filter is optimal with respect to all quadratic loss functions defining performance as a function of estimation error, but the measurement selection problem does not have that property. It depends very much on the particular loss function defining performance.

We present here a solution method based on what is called *maximum marginal benefit*. It is computationally efficient but suboptimal with respect to a given quadratic loss function of the resulting estimation errors $\hat{x} - x$:

$$\mathcal{L} = \sum_{\ell=1}^{N} ||A_\ell(\hat{x}_\ell(+) - x_\ell)||^2, \tag{8.166}$$

where the given matrices A_ℓ transform the estimation errors into other "variables of interest," as illustrated by the following examples:

1. If only the *final values* of the estimation errors are of interest, then $A_N = I$ (the identity matrix) and $A_\ell = 0$ (a matrix of zeros) for $\ell < N$.
2. If only a *subset of the state vector components* are of interest, then the A_ℓ will all equal the projection onto those components that are of interest.
3. If *any linear transformation* of the estimation errors is of interest, then the A_ℓ will be defined by that transformation.

4. If any *temporally weighted combination* of linear transformations of the estimation errors is of interest, then the corresponding A_ℓ will be the weighted matrices of those linear transformations. That is, $A_\ell = f_\ell B_\ell$, where $0 \leq f_\ell$ is the temporal weighting and the B_ℓ are the matrices of the linear transformations.

8.10.2 Marginal Optimization

The loss function is defined above as a function of the *a posteriori* estimation errors following measurements. The next problem will be to represent the dependence of the associated risk[8] function on the selection of measurements.

Parameterizing the Possible Measurements As far as the Kalman filter is concerned, a measurement is characterized by H (its measurement sensitivity matrix) and R (its covariance matrix of measurement uncertainty). A sequence of measurements is then characterized by the sequence

$$\{\{H_1, R_1\}, \{H_2, R_2\}, \{H_3, R_3\}, \ldots, \{H_N, R_N\}\}$$

of pairs of these parameters. This sequence will be called *marginally optimal* with respect to the above risk function if, for each k, the kth measurement is chosen to minimize the risk of the subsequence

$$\{\{H_1, R_1\}, \{H_2, R_2\}, \{H_3, R_3\}, \ldots, \{H_k, R_k\}\}.$$

That is, marginal optimization assumes that

1. the previous selections of measurements have already been decided.
2. no further measurements will be made after the current one is selected.

Admittedly, a marginally optimal solution is not necessarily a globally optimal solution. However, it does yield an efficient suboptimal solution method.

Marginal Risk Risk is the expected value of loss. The *marginal risk function* represents the functional dependence of risk on the selection of the kth measurement, *assuming that it is the last*. Marginal risk will depend only on the *a posteriori* estimation errors after the decision has been made. It can be expressed as an implicit function of the decision in the form

$$\mathcal{R}_k(P_k(+)) = E\left\{\sum_{\ell=k}^{N} ||A_\ell(\hat{x}_\ell(+) - x_\ell)||^2\right\}, \tag{8.167}$$

[8]The term *risk* is here used to mean the expected loss.

where $P_k(+)$ will depend on the choice for the kth measurement and, for $k < \ell \leq N$,

$$\hat{x}_{\ell+1}(+) = \hat{x}_{\ell+1}(-), \tag{8.168}$$

$$\hat{x}_{\ell+1}(+) - x_{\ell+1} = \Phi_\ell(\hat{x}_\ell(+) - x_\ell) - w_\ell, \tag{8.169}$$

so long as no additional measurements are used.

Marginal Risk Function Before proceeding with the development of a solution method, it will be necessary to derive an explicit representation of the marginal risk as a function of the measurement used. For that purpose, one can use a *trace formulation* of the risk function, as presented in the following lemma.

Lemma For $0 \leq k \leq N$, the risk function defined by Equation 8.167 can be represented in the form

$$\mathcal{R}_k(P_k) = \text{trace } \{P_k W_k + V_k\}, \tag{8.170}$$

where

$$W_N = A_N^{\mathrm{T}} A_N, \tag{8.171}$$

$$V_N = 0, \tag{8.172}$$

and, for $\ell < N$,

$$W_\ell = \Phi_\ell^{\mathrm{T}} W_{\ell+1} \Phi_\ell + A_\ell^{\mathrm{T}} A_\ell, \tag{8.173}$$

$$V_\ell = Q_\ell W_{\ell+1} + V_{\ell+1}. \tag{8.174}$$

Proof: A formal proof of the equivalence of the two equations requires that each be entailed by (derivable from) the other. We give a proof here as a reversible chain of equalities, starting with one form and ending with the other form. This proof is by backward induction, starting with $k = N$ and proceeding by induction back to any $k \leq N$. The property that the trace of a matrix product is invariant under cyclical permutations of the order of multiplication is used extensively.

INITIAL STEP The initial step of a proof by induction requires that the statement of the lemma hold for $k = N$. By substituting Equations 8.171 and 8.172 into Equation 8.170 and substituting N for k, one can obtain the following sequence of equalities:

$$\begin{aligned}
\mathcal{R}_N(P_N) &= \text{trace}\{P_N W_N + V_N\} \\
&= \text{trace}\{P_N A_N^{\mathrm{T}} A_N + 0_{n\times n}\} \\
&= \text{trace}\{A_N P_N A_N^{\mathrm{T}}\} \\
&= \text{trace}\{A_N E\langle(\hat{x}_N - x_N)(\hat{x}_N - x_N)^{\mathrm{T}}\rangle A_N^{\mathrm{T}}\}
\end{aligned}$$

$$
\begin{aligned}
&= \text{trace}\{E\langle A_N(\hat{x}_N - x_N)(\hat{x}_N - x_N)^{\mathrm{T}} A_N^{\mathrm{T}}\rangle\} \\
&= \text{trace}\{E\langle [A_N(\hat{x}_N - x_N)][A_N(\hat{x}_N - x_N)]^{\mathrm{T}}\rangle\} \\
&= \text{trace}\{E\langle [A_N(\hat{x}_N - x_N)]^{\mathrm{T}}[A_N(\hat{x}_N - x_N)]\rangle\} \\
&= E\langle ||A_N(\hat{x}_N - x_N)||^2\rangle.
\end{aligned}
$$

The first of these is Equation 8.170 for $k = N$, and the last is Equation 8.167 for $k = N$. That is, the statement of the lemma is true for $k = N$. This completes the initial step of the induction proof.

INDUCTION STEP One can suppose that Equation 8.170 is equivalent to Equation 8.167 for $k = \ell+1$ and seek to prove from this that it must also be the case for $k = \ell$. Then start with Equation 8.167, noting that it can be written in the form

$$
\begin{aligned}
\mathcal{R}_\ell(P_\ell) &= \mathcal{R}_{\ell+1}(P_{\ell+1}) + E\langle ||A_\ell(\hat{x}_\ell - x_\ell)||^2\rangle \\
&= \mathcal{R}_{\ell+1}(P_{\ell+1}) + \text{trace}\{E\langle ||A_\ell(\hat{x}_\ell - x_\ell)||^2\rangle\} \\
&= \mathcal{R}_{\ell+1}(P_{\ell+1}) + \text{trace}\{E\langle [A_\ell(\hat{x}_\ell - x_\ell)]^{\mathrm{T}}[A_\ell(\hat{x}_\ell - x_\ell)]\rangle\} \\
&= \mathcal{R}_{\ell+1}(P_{\ell+1}) + \text{trace}\{E\langle [A_\ell(\hat{x}_\ell - x_\ell)][A_\ell(\hat{x}_\ell - x_\ell)]^{\mathrm{T}}\rangle\} \\
&= \mathcal{R}_{\ell+1}(P_{\ell+1}) + \text{trace}\{A_\ell E\langle (\hat{x}_\ell - x_\ell)(\hat{x}_\ell - x_\ell)^{\mathrm{T}}\rangle A_\ell^{\mathrm{T}}\} \\
&= \mathcal{R}_{\ell+1}(P_{\ell+1}) + \text{trace}\{A_\ell P_\ell A_\ell^{\mathrm{T}}\} \\
&= \mathcal{R}_{\ell+1}(P_{\ell+1}) + \text{trace}\{P_\ell A_\ell^{\mathrm{T}} A_\ell\}.
\end{aligned}
$$

Now one can use the assumption that Equation 8.170 is true for $k = \ell+1$ and substitute the resulting value for $\mathcal{R}_{\ell+1}$ into the last equation above. The result will be the following chain of equalities:

$$
\begin{aligned}
\mathcal{R}_\ell(P_\ell) &= \text{trace}\{P_{\ell+1} W_{\ell+1} + V_{\ell+1}\} + \text{trace}\{P_\ell A_\ell^{\mathrm{T}} A_\ell\} \\
&= \text{trace}\{P_{\ell+1} W_{\ell+1} + V_{\ell+1} + P_\ell A_\ell^{\mathrm{T}} A_\ell\} \\
&= \text{trace}\{[\Phi_\ell P_\ell \Phi_\ell^{\mathrm{T}} + Q_\ell] W_{\ell+1} + V_{\ell+1} + P_\ell A_\ell^{\mathrm{T}} A_\ell\} \\
&= \text{trace}\{\Phi_\ell P_\ell \Phi_\ell^{\mathrm{T}} W_{\ell+1} + Q_\ell W_{\ell+1} + V_{\ell+1} + P_\ell A_\ell^{\mathrm{T}} A_\ell\} \\
&= \text{trace}\{P_\ell \Phi_\ell^{\mathrm{T}} W_{\ell+1} \Phi_\ell + Q_\ell W_{\ell+1} + V_{\ell+1} + P_\ell A_\ell^{\mathrm{T}} A_\ell\} \\
&= \text{trace}\{P_\ell [\Phi_\ell^{\mathrm{T}} W_{\ell+1} \Phi_\ell + A_\ell^{\mathrm{T}} A_\ell] + [Q_\ell W_{\ell+1} + V_{\ell+1}]\} \\
&= \text{trace}\{P_\ell [W_\ell] + [V_\ell]\},
\end{aligned}
$$

where Equations 8.173 and 8.174 were used in the last substitution. The last equation is Equation 8.170 with $k = \ell$, which was to be proved for the induction step. Therefore, by induction, the equations defining the marginal risk function are equivalent for $k \leq N$, which was to be proved.

IMPLEMENTATION NOTE The last formula separates the marginal risk as the sum of two parts. The first part depends only upon the choice of the measurement and the deterministic state dynamics. The second part depends only upon the stochastic state dynamics and is unaffected by the choice of measurements. As a consequence of this separation, the decision process will use only the first part. However, an assessment of the marginal risk performance of the decision process itself would require evaluation of the complete marginal risk function.

Marginal Benefit from Using a Measurement The *marginal benefit* resulting from the use of a measurement will be defined as the associated *decrease* in the marginal risk. By this definition, the marginal benefit resulting from using a measurement with sensitivity matrix H and measurement uncertainty covariance R at time t_k will be the difference between the *a priori* and *a posteriori* marginal risks:

$$\begin{aligned}
\mathcal{B}(H, R) &= \mathcal{R}_k(P_k(-)) - \mathcal{R}_k(P_k(+)) && (8.175)\\
&= \text{trace}\{[P_k(-) - P_k(+)]W_k\} && (8.176)\\
&= \text{trace}\{[P_k(-)H^{\mathrm{T}}(HP_k(-)H^{\mathrm{T}} + R)^{-1}HP_k(-)]W_k\} && (8.177)\\
&= \text{trace}\{(HP_k(-)H^{\mathrm{T}} + R)^{-1}HP_k(-)W_kP_k(-)H^{\mathrm{T}}\} && (8.178)
\end{aligned}$$

This last formula is in a form useful for implementation.

8.10.3 Solution Algorithm for Maximum Marginal Benefit

1. Compute the matrices W_ℓ using the formulas given by Equations 8.171 and 8.173.
2. Select the measurements in temporal order: for $k = 0, 1, 2, 3, \ldots, N$:
   (a) For each possible measurement, using Equation 8.178, evaluate the marginal benefit that would result from the use of that measurement.
   (b) Select the measurement that yields the *maximum* marginal benefit.

Again, note that this algorithm does *not* use the matrices V_ℓ in the trace formulation of the risk function. It is necessary to compute the V_ℓ only if the specific value of the associated risk is of sufficient interest to warrant the added computational expense.

TABLE 8.8 Complexity of Determining the Marginal Benefit of a Measurement

Operation	Complexity
$HP_k(-)$	$\mathcal{O}(\ell n^2)$
$HP_k(-)H^{\mathrm{T}} + R$	$\mathcal{O}(\ell^2 n)$
$[HP_k(-)H^{\mathrm{T}} + R]^{-1}$	$\mathcal{O}(\ell^3)$
$HP_k(-)W_k$	$\mathcal{O}(\ell n^2)$
$HP_k(-)W_kP_k(-)H^{\mathrm{T}}$	$\mathcal{O}(\ell^2 n)$
$\mathrm{trace}\{(HP_k(-)H^{\mathrm{T}} + R)^{-1}HP_k(-)W_kP_k(-)H^{\mathrm{T}}\}$	$\mathcal{O}(\ell^2)$
Total	$\mathcal{O}(\ell(\ell^2 + n^2))$

Note: ℓ is the dimension of the measurement vector; n is the dimension of the state vector.

8.10.3.1 Computational Complexity

Complexity of Computing the W_ℓ Complexity will depend upon the dimensions of the matrices A_ℓ. If each matrix A_ℓ is $p \times n$, then the products $A\ell^{\mathrm{T}} A_\ell$ require $\mathcal{O}(pn^2)$ operations. The complexity of computing $\mathcal{O}(N)$ of the W_ℓ will then be $\mathcal{O}(Nn^2(p + n))$.

Complexity of Measurement Selection The computational complexity of making a single determination of the marginal benefit of a measurement of dimension m is summarized in Table 8.8. On each line, the complexity figure is based on reuse of partial results from computations listed on lines above. If all possible measurements have the same dimension ℓ and the number of such measurements to be evaluated is μ, then the complexity of evaluating all of them[9] will be $\mathcal{O}(\mu\ell(\ell^2 + n^2))$. If this is repeated for each of $\mathcal{O}(N)$ measurement selections, then the total complexity will be $\mathcal{O}(N\mu\ell(\ell^2 + n^2))$.

8.11 INNOVATIONS ANALYSIS

Innovations are the differences between the expected measurements and the actual measurements. We have mentioned monitoring of the innovations to detect and reject anomalous measurement data.

The innovations can also be used as an indicator of the fidelity of the Kalman filter model being used. The information-weighted squared innovation

$$\chi_k^2 = (z_k - H_k\hat{x}_k(-))^T \left(H_kP_k(-)H_k^T + R_k\right)^{-1}(z_k - H_k\hat{x}_k(-)) \tag{8.179}$$

[9]Although the intermediate product $P_k(-)W_kP_k(-)$ [of complexity $\mathcal{O}(n^3)$] does not depend on the choice of the measurement, no reduction in complexity would be realized even if it were computed only once and reused for all measurements.

TABLE 8.9 Example Innovations Analysis of Mismodeled Implementations

Kalman Filter				Performance		
Parameter Values*				*A Priori*	*A Posteriori*	χ^2
Φ	H	Q	R	0.4813	0.2624	0.9894
2Φ	H	Q	R	1.4794	0.5839	3.5701
$\Phi/2$	H	Q	R	0.7237	0.4893	1.9269
Φ	$2H$	Q	R	0.6762	0.5938	0.3843
Φ	$H/2$	Q	R	0.9637	0.9604	1.9526
Φ	H	$2Q$	R	0.6133	0.5015	0.6411
Φ	H	$Q/2$	R	0.6036	0.4860	1.4248
Φ	H	Q	$2R$	0.6036	0.4861	0.7130
Φ	H	Q	$R/2$	0.6133	0.5015	1.2810

*Simulated process model parameters are Φ, H, Q, and R.

has a theoretical mean value of 1 when the problem is perfectly modeled. Deviations from that value may indicate mismodeling problems.

As an illustration of that approach, the enclosed MATLAB file `InnovAnalysis.m` simulates a linear stochastic process with nine different Kalman filters: one with the same model parameters as the simulated process and eight with the four model parameters Φ, H, Q, and R scaled up and down by a factor of 2. The empirical mean values of the χ^2 statistics are then compared. In addition, because the process is being simulated, one can calculate the simulated filter performances in terms of the RMS differences between the simulated state variable and the estimates from the nine Kalman filters.

The results of a single Monte Carlo simulation with 1000 time steps are summarized in Table 8.9. These demonstrate that the correctly modeled Kalman filter has a χ^2 statistic close to 1 and significant deviations of the same χ^2 statistics for the eight different modeling deviations. Some mean χ^2 values are larger than those from the correctly modeled case, and some are smaller. In all cases, the *a priori* and *a posteriori* RMS estimation accuracies of the mismodeled Kalman filters implementations are worse than those of the correctly modeled Kalman filter—but one cannot know the true state variables except in simulation. The χ^2 statistics, on the other hand, can always be calculated from the Kalman filter parameter values and the filter innovations.

8.12 SUMMARY

This chapter has discussed methods for the design and evaluation of estimation systems using Kalman filters. Specific topics addressed include the following:

1. methods for detecting and correcting anomalous behavior of estimators,
2. predicting and detecting the effects of mismodeling and poor unobservability,

3. evaluation of suboptimal filters (using dual-state filters) and sensitivity analysis methods,
4. comparison of memory, throughput, and worldlength requirements for alternative implementation methods,
5. methods for decreasing computational requirements,
6. methods for assessing the influence on estimator performance of sensor location and type and the number of sensors,
7. methods for top-down hierarchical system-level error budgeting, and
8. demonstration of the application of square-root filtering techniques to an INS-aided GPS navigator.

PROBLEMS

8.1 Show that the final value of the risk obtained by the marginal optimization technique of Section 8.10 will equal the initial risk minus the sum of the marginal benefits of the measurements selected.

8.2 Develop the equations for dual-state error propagation by substituting Equations 8.90 and 8.91 into Equations 8.94 and 8.95, using Equation 8.97 explicitly.

8.3 Obtain the dual-state vector equation for the covariances of the system and error, where x_1 is a ramp plus random walk and x_2 is constant:

$$\dot{x}_1^S = x_2^S + w^S, \quad \dot{x}_2^S = 0, \quad z_k = x_k^1 + v_k,$$

using as the filter model a random walk

$$\dot{x}^F = w^F, \quad z_K = x_k^F + v_k.$$

8.4 Derive the results of Example 8.4.

8.5 Prove that $cov[\tilde{x}_k^S]$ depends upon $cov(x_k^S)$

8.6 Rework Problem 4.6 for the UDU^{T} formulation and compare your results with those of Problem 4.6.

8.7 Rework Problem 4.7 for the UDU^{T} formulation and compare your results with those of Problem 4.7.

8.8 Solve Problem 4.6 using the Schmidt–Kalman filter (Section 8.6) and compare the results with those of Example 4.4.

9

APPLICATIONS TO NAVIGATION

"Scientists discover the world that exists; engineers create the world that never was."
—Theodore von Kármán (1881–1963)

It is hard to imagine an application for Kalman filtering more fruitful than navigation. Before Kalman filtering, navigation was practiced only by highly trained technical specialists. Today, it is available as a commodity service for the consumer, requiring no more skill than that needed to use a telephone. In this chapter, we present some of the technical details behind automated navigation systems.

9.1 CHAPTER FOCUS

9.1.1 Navigation

Navigation Problem The navigation problem is to direct the movement of a vehicle (e.g., a wheeled vehicle, watercraft, aircraft, or spacecraft) so as to arrive at a given destination. An important part of navigation is determining one's location relative to one's destination and relative to local features related to travel (roads, canals, shipping lanes, etc.).

Navigation Solution The solution to the navigation problem generally requires observations or measurements of some kind and the ability to use that information

Kalman Filtering: Theory and Practice using MATLAB®, Third Edition. By Mohinder S. Grewal and Angus P. Andrews

to determine your location relative to your destination. The Kalman filter has played a major role in solving the navigation problem.

There are five basic forms of navigation:

1. Pilotage, which essentially relies on recognizing landmarks to know where you are and how you are oriented. It is older than humankind.
2. Dead reckoning, which relies on knowing where you started from, plus some form of heading information (e.g., a magnetic compass) and some estimate of the distance traveled.
3. Celestial navigation, using time and the angles between local vertical and known celestial objects (e.g., the sun, moon, planets, stars) to estimate orientation, latitude and longitude.
4. Radio navigation, which relies on radio-frequency sources with known locations. Global navigation satellite systems (GNSS) use beacons on satellites for that purpose.
5. Inertial navigation, which relies on knowing your initial position, velocity, and attitude and thereafter measuring your attitude rates and accelerations. It is the only form of navigation that does not rely on external references.

This chapter focuses on emerging applications of Kalman filtering in solving the navigation problem, especially combinations of the last two methods listed above. The development of low-cost receivers for GNSS and low-cost micro-electro-mechanical systems (MEMS) technologies for inertial navigation systems (INS) have revolutionized the potential cost/performance ratios for high-accuracy navigation.

INS provides excellent short-term navigation solutions, the accuracy of which degrades over time. GNSS provides coarser short-term solutions which do not degrade over time. When these two navigation systems are integrated using a Kalman filter, the resulting navigation system outperforms either of its component subsystems—even when inertial sensors with lower performance and cost are used. Decreasing costs are opening up many new applications markets for integrated GNSS/INS systems, which require Kalman filter implementations tailored to these applications.

This chapter is about the design and implementation of Kalman filters for these applications, including derivations of models used with the associated Riccati equations for predicting the performance of potential sensor system designs. These include examples of Kalman filter models for GNSS and INS errors and practical Kalman filter implementation architectures for integrated navigation solutions.

9.1.2 Kalman Filtering in Navigation

As we mentioned in Chapter 1, the first practical application of Kalman filtering was for estimating the n-body space trajectories of the Apollo spacecraft from the earth to the moon and back. This was a rather difficult estimation problem at that time, because the dynamic model (presented in Example 2.1) is time-varying and nonlinear, and the accuracy requirements for predicting and controlling terminal conditions at

the moon and on earth were quite severe. Had the Kalman filter not appeared when it did, it is questionable whether the project would have succeeded as well as it did.

Since that time, the Kalman filter has come to play a major role in the development of advanced navigation systems, including GNSS, INS, and combinations of the two.

9.1.3 Main Points to Be Covered

Kalman Filter Models for Navigation Kalman filter models used in GNSS/INS navigation represent:

1. *The deterministic and nondeterministic dynamics of the vessel or vehicle that is to be located and tracked as part of the navigation problem*. It is often called the *host vehicle*, because it hosts the navigation sensors to be used. It could be a spacecraft, aircraft, surface ship, submarine, land vehicle, or pack animal (nonhuman or human). The motions of the host vehicle are the fundamental dynamics of the navigation problem, and the dynamic model for the host vehicle is an important part of the overall Kalman filter model used for optimizing navigation.
2. *Error characteristics of INS*. These include error sources in the inertial sensors, the mechanisms by which they cause navigation errors, and the dynamics of the resulting errors in derived estimates of position, velocity, and attitude.
3. *GNSS receiver error sources and their statistical characteristics*. These include error sources within the receiver, such as internal clock errors. They also include GNSS system-wide errors "upstream" of the receiver, such as randomly varying atmospheric propagation delays, background electromagnetic noise, or satellite position errors.
4. *Serial processing of pseudorange measurements*. A pseudorange

$$\rho = c \times (t_{\text{receive}} - t_{\text{transmit}})$$

   is the apparent distance an electromagnetic signal has traveled if it was transmitted at time t_{transmit} and received at time t_{receive}. The parameter c is the estimated speed of transmission, which varies ever so slightly in the atmosphere—from time to time and location to location. Errors in GNSS pseudorange measurements to different satellites are uncorrelated, which allows each pseudorange to be processed independently as a scalar measurement. This avoids matrix inversion in the Kalman filter implementation, which reduces the amount of computation as well as the effects of roundoff errors.
5. *Synergistic relationships between the GNSS and INS*. These include such things as the use of inertial sensor outputs to enhance GNSS signal phase-lock, and the use of GNSS aiding to estimate and correct for INS sensor errors.

All of these models are required for integrating GNSS and a given INS on a particular host vehicle.

These models may also be used in the design of an INS for a particular set of trajectories in a particular host vehicle. In this case, parametric INS models are used to represent the possible choices of inertial sensors in the INS. In addition to achieving a desired integrated GNSS/INS navigation performance, objectives of the design problem may include such additional performance features as

1. Minimizing the cost of the onboard integrated GNSS/INS system;
2. Achieving specified stand-alone inertial navigation performance when GNSS signals are lost;
3. Reduction of GNSS signal reacquisition time after loss of the signal; and
4. Improving GNSS signal lock-on during severe dynamic maneuvers of the host vehicle.

Kalman Filter Implementations We also provide examples of alternative Kalman filter implementations for GNSS and INS navigation, including

1. Stochastic system models for INS navigation errors. These can be used to augment INS navigation with auxiliary sensor systems, including GNSS.
2. Stand-alone GNSS navigation, used for
   (a) GNSS receivers on host vehicles.
   (b) Stationary GNSS receivers for estimating the location of the receiver antenna in earth-fixed coordinates, or to provide accurate timing, or both.
3. Integrated GNSS/INS navigation systems. Practical implementation architectures depend on
   (a) What types of data can be obtained from the GNSS receiver and the INS for use in the Kalman filter.
   (b) What internal error sources within the GNSS receiver and/or the INS can be compensated using the Kalman filter solution.

   The phrase *loosely coupled* is used to denote integration architectures using only the standard GNSS and INS outputs from their stand-alone configurations. The phrase *tightly coupled* is used for those architectures requiring access to and control of internal intermediate data values and/or control variables in the GNSS receiver or INS navigation computer.
4. INS navigation using other auxiliary sensor systems as navigation aids. These may include
   (a) Position sensors such as
      i. Radio navigation aids (e.g., LORAN, OMEGA, VOR, DME).
      ii. Radar (onboard and/or ground-based).
      iii. Laser ranging to known landmarks.
      iv. Terrain matching (radar and/or passive imaging).
      v. Barometric altimeters.
      vi. Radar and/or laser altimeters.
      vii. Depth (pressure) sensors—for submersibles.

(b) Velocity sensor such as
   i. Doppler radar.
   ii. Airspeed sensors.
   iii. Waterspeed sensors (electromagnetic log).
(c) Attitude sensors such as
   i. Attitude and heading reference systems (AHRS).
   ii. Star trackers.
   iii. Interferometric multiantenna GNSS attitude sensors.

This chapter focuses primarily on Kalman filter models and their implementations. For more historical background on the subject, see, for example, [174]. For broader technical coverage of the subject, see, for example, [106]. For more in-depth coverage of inertial navigation, see, for example, [166, 270, 283]. For better coverage of GNSS receiver technology, see, for example, [271]. Because GNSS and INS technologies are still evolving at a significant rate, technical journals and periodicals on the subject are also good sources for the latest developments.

9.2 HOST VEHICLE DYNAMICS

9.2.1 Host Vehicles

An important part of Kalman filter models used in navigation is a model for the dynamic behavior of the vehicle which is to be tracked during its journey, often referred to as the *host vehicle*. However, in applications such as air defense[1] or air traffic control, the only navigation sensors may be radar systems that are not hosted on the vehicle being tracked.

9.2.2 Dynamic Models

Host vehicle dynamic models represent the likelihood of potential maneuvers of the host vehicle, given measurements of its dynamic conditions (position and/or velocity, mostly). Good modeling takes advantage of any constraints on the mobility of the host vehicle. If the host vehicle is a ship at sea, for example, its capabilities for vertical motion are essentially nill and its lateral maneuvering capabilities are very limited. In similar fashion, railroad trains are confined to run on existing tracks, and highway vehicles are primarily restricted to available roadways and facilities with paved access thereto.

9.2.2.1 Mathematical Forms Table 9.1 list some of the parametric models used for modeling the motion of a host vehicle. These are specified in terms of the dynamic coefficient matrix F and the disturbance noise covariance Q for a single axis of

[1]Perhaps reflecting their heritage in air defense, some civilian ground-based radar tracking implementations still use the term *target* for objects being tracked.

motion. Depending on the application, the host vehicle dynamic model may have different models for different axes. Surface ships, for example, may assume constant altitude and only estimate the north and east axes of position and velocity.

9.2.3 Determining Dynamic Model Parameters

The model parameters listed in Table 9.1 include standard deviations σ and correlation time constants τ of position, velocity, acceleration, and jerk (derivative of acceleration). Those parameters labeled as "independent" can be specified by the designer. Those labeled as "dependent" will depend on the values specified for the independent variables. (See [106] for more details.)

The choice of a model and the values of its parameters will depend on the dynamic capabilities of the host vehicle in the navigation problem being considered.

9.2.3.1 Stationary Model The first model in Table 9.1 is for an object fixed to the earth, such as a GNSS antenna at a fixed location or an INS under test in a laboratory

TABLE 9.1 Host Vehicle Dynamic Models

Mod. No.	Model Parameters (Each Axis) F	Model Parameters (Each Axis) Q	Indep. Param.	Depend. Param.
1	0	0	None	None
2	$\begin{bmatrix} 0 & 1 \\ -\dfrac{\sigma_{\text{vel}}^2}{\sigma_{\text{pos}}^2} & \dfrac{-2\,\sigma_{\text{vel}}}{\sigma_{\text{pos}}} \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 0 & \dfrac{4\,\sigma_{\text{vel}}^3}{\sigma_{\text{pos}}} \end{bmatrix}$	σ_{pos}^2 σ_{vel}^2	δ (damping) σ_{acc}^2
3	$\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 0 & \sigma_{\text{acc}}^2\,\Delta t^2 \end{bmatrix}$	σ_{acc}^2	$\sigma_{\text{pos}}^2 \to \infty$ $\sigma_{\text{vel}}^2 \to \infty$
4	$\begin{bmatrix} 0 & 1 \\ 0 & -1/\tau_{\text{vel}} \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 0 & \sigma_{\text{acc}}^2\,\Delta t^2 \end{bmatrix}$	σ_{vel}^2 τ_{vel}	$\sigma_{\text{pos}}^2 \to \infty$ σ_{acc}^2
5	$\begin{bmatrix} 0 & 1 & 0 \\ 0 & \dfrac{-1}{\tau_{\text{vel}}} & 1 \\ 0 & 0 & \dfrac{-1}{\tau_{\text{acc}}} \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \sigma_{\text{jerk}}^2\,\Delta t^2 \end{bmatrix}$	σ_{vel}^2 σ_{acc}^2 τ_{acc}	$\sigma_{\text{pos}}^2 \to \infty$ τ_{vel} $\rho_{\text{vel, acc}}$ σ_{jerk}^2
6	$\begin{bmatrix} \dfrac{-1}{\tau_{\text{pos}}} & 1 & 0 \\ 0 & \dfrac{-1}{\tau_{\text{vel}}} & 1 \\ 0 & 0 & \dfrac{-1}{\tau_{\text{acc}}} \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \sigma_{\text{jerk}}^2\,\Delta t^2 \end{bmatrix}$	σ_{pos}^2 σ_{vel}^2 σ_{acc}^2 τ_{acc}	τ_{pos} τ_{vel} $\rho_{\text{pos, vel}}$ $\rho_{\text{pos, acc}}$ $\rho_{\text{vel, acc}}$ σ_{jerk}^2

environment. In these cases, the parameters $F = 0$ and $Q = 0$, and the host vehicle could be a building. This model would also apply when GNSS is used in surveying for determining the location of a stationary antenna—usually with respect to another antenna at a known fixed location.

This model has no parameters to adjust.

9.2.3.2 Quasistationary Model The second model in Table 9.1 is for critically damped harmonic motion with random acceleration excitation. It is used as a model for a host vehicle nominally stationary, but with limited dynamic disturbances. This would apply to ships tied up at dockside and to aircraft or land vehicles parked during fueling and loading operations. This type of model is used for INS during initial alignment operations on a quasistationary host vehicle, when the inertial sensors can detect small, uncontrolled disturbances in acceleration and rotation. A similar model may apply to the vertical dynamics of ships at sea.

The independent parameters in this model are RMS position excursion σ_{pos} and RMS velocity σ_{vel}, which would ordinarily be determined from empirical data taken onboard the host vehicle during normal operations. These parameters are related to the critical damping factor δ and mean-squared acceleration excitation σ_{acc}^2 by the equations

$$\delta = \frac{2\,\sigma_{\text{vel}}}{\sigma_{\text{pos}}} \tag{9.1}$$

$$\sigma_{\text{acc}}^2 = \frac{4\,\sigma_{\text{vel}}^3}{\sigma_{\text{pos}}}. \tag{9.2}$$

The units of σ_{acc}^2 in this case are squared acceleration divided by time, which is consistent with the model in continuous time. The equivalent value in discrete time will depend on the discrete time interval Δt and can usually be approximated as $\Delta t \times \sigma_{\text{acc}}^2$.

This is a good model for the adaptive suspension systems found in aircraft and some trucks, which maintain the vehicle at the same height as it is loaded and unloaded. Some ships use ballast for this purpose.

The critically damped vehicle suspension system model can also be generalized to an overdamped suspension system or an underdamped suspension system with known resonant frequency and damping. Similar models can be used for rotational dynamics, which are important for gyrocompass alignment of INS.

9.2.3.3 Type 2 Tracker The third model in Table 9.1 is one of the most commonly used in navigation. It is used for tracking GNSS receiver phase and frequency error, and is often used for tracking the position and velocity of a host vehicle.

The only adjustable parameter is the mean-squared acceleration noise σ_{acc}^2. In the equivalent discrete-time model, a value for disturbance noise covariance can sometimes be determined empirically as the RMS velocity change over the sample interval.

9.2.3.4 Modified Type 2 Tracker The fourth model in Table 9.1 is a refinement of the type 2 tracker for vehicles with bounded velocity capability. These trackers can perform better when GNSS signals are lost.

The adjustable parameters in this case include the mean-squared velocity and the velocity correlation time, which can often be determined empirically.

9.2.3.5 Model for Bounded RMS Velocity and Acceleration The fifth model is a further refinement for vehicles with bounded velocity and acceleration. These also perform better when signals are lost, because they take into account the true limitations of the host vehicle.

The parameter values (mean-squared velocity and acceleration and the acceleration correlation time) can often be determined empirically. Instrumentation for sampling the empirical data may include three-axis accelerometer clusters or an INS.

9.2.3.6 Models for Bounded RMS Position The last model in Table 9.1 is for vehicles with bounded position as well. This model may be appropriate for the limited altitude dynamics of land vehicles, for example, or for tidal effects on the altitudes of ships.

Alternative Control-Based Model There is a related control model for bounding the RMS position, velocity, and acceleration of a vehicle disturbed by white jerk noise:

$$\frac{d}{dt}\begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ f_{3,1} & f_{3,2} & f_{3,3} \end{bmatrix}\begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ w_{\text{jerk}}(t) \end{bmatrix} \tag{9.3}$$

$$f_{3,1} = -\frac{\sigma_{\text{jerk}}^2 \sigma_{\text{vel}}^2}{2(\sigma_{\text{pos}}^2 \sigma_{\text{acc}}^2 - \sigma_{\text{vel}}^4)} \tag{9.4}$$

$$f_{3,2} = -\frac{\sigma_{\text{acc}}^2}{\sigma_{\text{vel}}^2} \tag{9.5}$$

$$f_{3,3} = -\frac{\sigma_{\text{pos}}^2 \sigma_{\text{jerk}}^2}{2(\sigma_{\text{pos}}^2 \sigma_{\text{acc}}^2 - \sigma_{\text{vel}}^4)} \tag{9.6}$$

$$\sigma_{\text{jerk}}^2 = \mathop{\text{E}}_{t}\left\langle w_{\text{jerk}}^2(t) \right\rangle \tag{9.7}$$

$$= \text{mean-squared jerk noise}$$

$$\sigma_{\text{pos}} = \text{RMS position excursion}$$

$$\sigma_{\text{vel}} = \text{RMS velocity}$$

$$\sigma_{\text{acc}} = \text{RMS acceleration}$$

$$\sigma_{\text{pos}}^2 \sigma_{\text{acc}}^2 > \sigma_{\text{vel}}^4. \tag{9.8}$$

This can be verified by solving the associated steady-state Riccati equation,

$$0 = FP + PF^T + Q \tag{9.9}$$

$$Q = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \sigma^2_{\text{jerk}} \end{bmatrix} \tag{9.10}$$

for P, giving the diagonal elements as the steady-state σ^2_{pos}, σ^2_{vel}, and σ^2_{acc}. The result can be solved for F as a function of σ^2_{pos}, σ^2_{vel}, σ^2_{acc}, and σ^2_{jerk}.

Empirical Modeling For most host vehicles, the best instrumentation for determining the required dynamic model statistics is an INS with a data recorder.

9.3 INERTIAL NAVIGATION SYSTEMS (INS)

Overview Kalman filters for inertial navigation applications use fairly sophisticated and faithful models for the dynamics of inertial navigation errors. When INSs are integrated with other navigation aids, these models provide opportunities for exploiting statistical differences between the various error sources. The combination of these models with Kalman filtering has enabled greatly improved navigation accuracies at significantly lower costs than had been attainable before.

The general approach is state vector augmentation to accommodate nonwhite noise in the host vehicle state measurements from the INS, but now the time-correlated noise models are far more complex than those for exponentially correlated noise. The INS measurement z_{INS} of the host vehicle state vector x_{NAV} can be modeled as

$$\begin{aligned} \dot{x}_{\text{NAV}} &= F_{\text{NAV}} x_{\text{NAV}} + w_{\text{NAV}}(t) \\ z_{\text{INS}} &= H_{\text{INS}} x_{\text{NAV}} + \varepsilon_{\text{INS}} + v_{\text{INS}}(t) \\ \dot{\varepsilon}_{\text{INS}} &= F_{\text{INS}} \varepsilon_{\text{INS}} + w_{\text{INS}}(t), \end{aligned}$$

where x_{NAV} is the host vehicle navigation state variable, the dynamic coefficient matrix F_{NAV} models host vehicle dynamics, ε_{INS} is the vector of INS errors, and F_{INS} is its dynamic coefficient matrix.

In state vector augmentation, the INS error vector ε_{INS} is appended to the navigation state vector x_{NAV}

$$\frac{d}{dt}\begin{bmatrix} x_{\text{NAV}} \\ \varepsilon_{\text{INS}} \end{bmatrix} = \begin{bmatrix} F_{\text{NAV}} & 0 \\ 0 & F_{\text{INS}} \end{bmatrix}\begin{bmatrix} x_{\text{NAV}} \\ \varepsilon_{\text{INS}} \end{bmatrix} + \begin{bmatrix} w_{\text{NAV}}(t) \\ w_{\text{INS}}(t) \end{bmatrix}$$

to obtain a combined system model for the host vehicle and its INS.

The INS measurements can also be combined with auxiliary measurements of the host vehicle dynamics,

$$z_{\mathrm{AUG}} = \begin{bmatrix} z_{\mathrm{AUX}} \\ z_{\mathrm{INS}} \end{bmatrix} = \begin{bmatrix} H_{\mathrm{AUX}} & 0 \\ H_{\mathrm{INS}} & I \end{bmatrix} \begin{bmatrix} x_{\mathrm{NAV}} \\ \varepsilon_{\mathrm{INS}} \end{bmatrix} + \begin{bmatrix} v_{\mathrm{AUX}} \\ v_{\mathrm{INS}} \end{bmatrix},$$

where z_{AUX} is the auxiliary measurement vector and H_{AUX} is its measurement sensitivity matrix.

This model can be further complicated by time-correlated dynamic disturbance noise $w_{\mathrm{INS}}(t)$ driving the INS navigation error state $\varepsilon_{\mathrm{INS}}$, which calls for an additional round of state vector augmentation. Examples are developed and demonstrated in the following subsections.

9.3.1 Inertial Navigation

Inertial navigation had its beginnings in World War II with German cruise missiles (V-1) and short-range ballistic missiles (V-2). It was the first successful long-range autonomous navigation method, requiring no operator actions once it was underway. It does not rely on any external infrastructure, it cannot be jammed or spoofed, and it is inherently stealthy. It would become the most common navigation method for military and commercial aviation. It has remained the navigation method of choice for applications that cannot rely on GNSS (e.g., submarines and other strategic weapons platforms).

9.3.1.1 Inertial Navigation Theory The fundamental idea behind inertial navigation is that the second integral of acceleration is position. Inertial navigation begins with known initial values for position and velocity. Thereafter, acceleration is measured and integrated twice to obtain the current position. (In practice, it is a bit more complicated.)

9.3.1.2 Practical Inertial Navigation An inertial navigation system commonly includes:

1. An inertial sensor assembly (ISA) for measuring *translational accelerations* (three components) and *rotation rates* (three components). It includes *accelerometers* (acceleration sensors) and *gyroscopes* (attitude rate sensors), all rigidly affixed to a common base.
2. Digital processing systems (one or more) for
   (a) Processing sensed rotation rates to maintain knowledge of the *attitude* of the inertial sensors relative to a reference coordinate frame. The choice of coordinates is usually driven by the application. For navigation relative to the earth, it is usually earth-fixed. For surface navigation, it may have locally level coordinates (e.g., east-north-up).

(b) Calculating and compensating for any rotation of the coordinate frame (earth rotation, for example).

(c) Calculating the local gravitational acceleration, which is not detectable by the accelerometers, and adding it to the sensed acceleration to determine net acceleration.

(d) Integrating accelerations twice—first, to determine changes in vehicle *velocity*, and again, to determine changes in *position*.

(e) Compensating the sensor outputs for known deviations in sensor scale factors, biases (output offsets), input axis misalignments, etc.

(f) Performing other operational functions such as initial alignment (estimating attitude) and processing inputs and outputs (including driving any displays).

3. Supporting electromechanical subsystems for such functions as
   (a) Operator interfaces, including displays.
   (b) Electrical power conversion and conditioning.
   (c) Temperature regulation (if required).
   (d) Signal interfaces to other onboard systems.
4. Mechanical isolation from shock and vibration transmitted through the vehicle mounting location. In what is known as a *strapdown configuration*, this is the only form of mechanical isolation. In what is known as a *gimbaled configuration*, mechanical isolation also includes isolation of the sensors from rotations of the host vehicle.

Gimbaled INS The first standalone INSs were gimbaled. Their attitude isolation is either from a spherical bearing or a set of gimbal rings. Gyroscopes in feedback control servo loops generate torques to null out any sensed attitude changes in the inertial sensor assembly.

Strapdown INS Strapdown systems use software to replace gimbals. They compute the coordinate transformations needed to resolve accelerations and attitude rates from sensor-fixed coordinates to the coordinates of the navigation problem. Figure 9.1 shows the major computational processes involved when strapdown navigation is performed using latitude, longitude, and altitude to represent the vehicle's location. As part of the navigation solution, these processes can also generate many other variables useful in navigation, such as the heading, pitch, and roll angles of the host vehicle.

The computational requirements for strapdown operation are such that the technology could not be used until there were digital processors capable of performing attitude and acceleration integrations fast enough (typically on the order of 100 times per second) to maintain navigation accuracy but suitable for mobile applications.

In the late 1960s, the Apollo Lunar Excursion Module used strapdown systems as backup navigators for transporting two astronauts from lunar orbit to the surface of the moon and back. Today, strapdown systems dominate many inertial navigation markets and generally cost less than gimbaled systems.

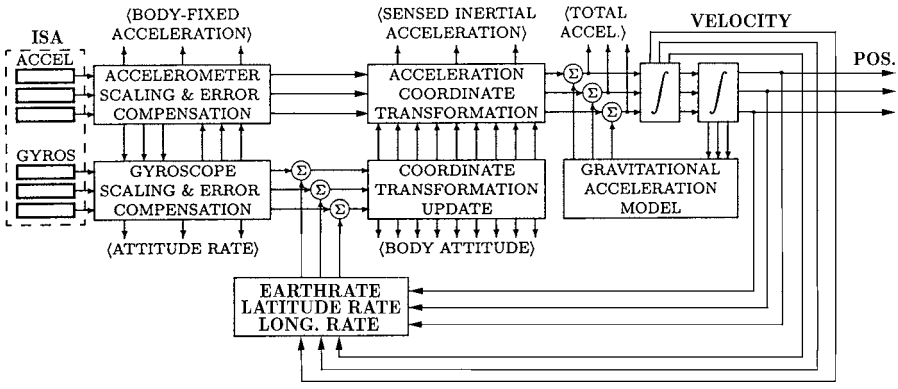


Figure 9.1 Major computational processes in strapdown INS.

9.3.2 Initializing the Navigation Solution

9.3.2.1 Calibration and Alignment These procedures are usually performed prior to navigation, but they can also be performed during navigation if auxiliary sensor inputs (e.g., from GNSS) are available. Both procedures are commonly implemented as Kalman filters.

Sensor calibration is the process of estimating the relationship between the inputs and outputs of sensors. It is usually implemented by a procedure in which inputs are controlled and outputs are recorded, and then these data are used (in a Kalman filter, typically) to model the input/output relationship. These are usually parametric models, and Kalman filtering is often used for estimating the best-fit parameters.

The purpose of calibration is to be able to convert sensor outputs to equivalent inputs so that an output value (either analog or digital) can be converted back to input acceleration units (e.g., meters per second squared) or input rotation rate units (e.g., radians per second) for use in navigation.

Calibration is used to compensate for constant, correctable error sources in sensors, including such attributes as

1. Variations in the sensitive axis direction, which determines which component of acceleration or rotation the sensor measures.
2. Variations in the sensor *scale factor*, which is the ratio of output variation to input variation.
3. *Sensor bias*, which is the output offset when there is no input.
4. Temperature sensitivity in the case where temperature compensation is required.

The calibration parameters may also model such effects as nonlinearity, acceleration sensitivity of gyroscopes, or attitude rate sensitivities of accelerometers.

Alignment is the process of determining the orientation of the ISA with respect to navigation coordinates.

Gyrocompass alignment is a method for determining the initial orientation of the ISA with respect to earth-fixed coordinates. This is normally done when the INS is quasistationary, in that any disturbances of attitude and position are small, cyclical, and essentially zero-mean. This model would apply, for example, to conditions aboard an aircraft parked for loading and refueling or to a ship tied up at dockside.

For alignment in earth-fixed coordinates, sensed accelerometers are used to measure the direction of gravity with respect to the ISA, and the gyroscopes measure the direction of the earth rotation axis with respect to the ISA. These two directions in ISA-fixed coordinates are sufficient to determine the orientation of the ISA with respect to the earth—except when the alignment is done near the North and South Poles. Using the directions of gravity and the earth's rotation axis to determine orientation is called *gyrocompassing*. When these two reference directions (gravity and the earth rotation axis) coincide (as they do at the poles), gyrocompassing is not sufficient for determining orientation.

Gyrocompass alignment normally uses a Kalman filter with a damped harmonic oscillator model for disturbances of the host vehicle during the alignment period. The modeled resonant frequency and damping time constant are generally determined empirically or by matching them to the host vehicle suspension system. The covariance of dynamic disturbance noise is best determined empirically.

9.3.2.2 Initializing Position and Velocity In "pure" inertial navigation (i.e., without aiding sensors), one must provide initial values for integrating accelerations to determine velocity and for integrating velocity to determine position.

Initial Position Many navigation problems start with the host vehicle parked in a fixed location, which is usually known. Although gyrocompass alignment estimates the initial latitude of the INS, it is common practice to use external sources—including GNSS—for initializing longitude, latitude, and altitude. However, in integrated GNSS/INS implementations it is possible to estimate attitude, position, and velocity when the host vehicle is underway.

Initial Velocity For parked host vehicles, this is zero. In integrated GNSS/INS implementations, it can be estimated when the host vehicle is underway.

9.3.3 Cruise Navigation Coordinates

Navigation coordinates are the reference frame in which the navigation problem is to be solved.

9.3.3.1 Cruise Navigation The most common applications of navigation are for travel on or near the surface of the earth, such as for navigation of aircraft, ships, and land vehicles. These are collectively called *cruise applications* of navigation or *cruise navigation*.

In most cruise applications of inertial navigation, the dominant sensed acceleration is that required to counter gravity ($\approx$1 g straight up). This has a strong influence on the dynamics of INS errors.

9.3.3.2 Locally Level Coordinates Most cruise navigation problems are solved using latitude, longitude, and altitude for specifying location and a *locally level* coordinate system for specifying directions at that location. Local directions are usually referenced to the north, east, and vertical, using Cartesian coordinates with these reference axes.

9.3.3.3 North-East-Down (NED) Coordinates It is fairly common in inertial navigation to make the vertical reference axis point down. For a vehicle that is level and headed north, this convention makes the local north, east, and down axes parallel to the vehicle roll, pitch, and yaw axes, respectively, and it makes the measured heading angle (rotation about the vertical axis) be clockwise from north. In early gimbaled INS with three gimbal rings, the gimbal system was designed such that the three gimbal angles were the vehicle heading, pitch, and roll angles.

9.3.4 Inertial Navigation Solution

The navigation solution for INS includes the position, velocity, and orientation (or attitude) of the host vehicle. Outputs may also include accelerations and angular rates, which are useful for automatic control and guidance of the host vehicle.

9.3.4.1 Velocity and Position Solution Accelerations sensed in the ISA are resolved into navigation coordinates, where they are integrated once to compute velocity increments and again to compute position increments.

9.3.4.2 Orientation/Attitude Solution Attitude rates sensed in the ISA are used to maintain knowledge of the orientation of the ISA with respect to navigation coordinates so that this knowledge can be used for resolving sensed accelerations into navigation coordinates.

In gimbaled systems, the sensed attitude rates are used in feedback servo loops to generate gimbal torques that will maintain the ISA aligned to locally level coordinates. Inertial rotation rates of locally level coordinates (due to earth rotation and the rates of change in vehicle latitude and longitude) are injected into the servo loops to maintain the ISA level and pointed north.

In strapdown systems, the gimbals are replaced by a coordinate transformation from ISA coordinates to navigation coordinates. Sensed rotation rates and coordinate rotation rates are used in computer algorithms to maintain the correct value of the coordinate transformation.

Attitude integration is noncommutative, which has created its share of difficulties in strapdown navigation. Rotating 30° about the east axis and then 20° about the north axis is not the same as rotating 20° about the north axis and then 30° about the east axis.

Among the many different approaches to solving the problem, the most successful has been to represent attitude on the unit sphere of quaternions and apply a formula derived by Bortz [43] for reducing the required integration sample rates. See, for example, [270] for details.

9.3.4.3 Gravity Modeling Accelerometers cannot measure acceleration due to gravity. An INS at rest on the surface of the earth can sense the earth pushing it up but cannot sense gravity pulling it down. To correct for this, gravitational accelerations must be computed (using a gravity model) and added to the sensed acceleration to obtain net acceleration. Figure 9.1 shows how a gravity model is used in strapdown inertial navigation. Position information is used in computing the local gravitational acceleration, which is subtracted from sensed acceleration to compute acceleration with respect to earth-fixed coordinates.

Gravity models required for inertial navigation vary with performance requirements. High-accuracy inertial navigators have performance requirements on the order of 0.1 nautical mile per hour CEP rate,[2] which requires very sophisticated gravity modeling. These models generally include very-high-order spherical harmonics of vertical deviations of equipotential surfaces.

INS applications with less stringent performance requirements require less sophisticated gravity models, and systems integrated with GNSS require even less accurate gravity models.

9.3.4.4 Earthrate Modeling Because the earth rotates, all earth-fixed coordinate systems rotate with it. The inertial navigation solution in earth-fixed coordinates requires earthrate modeling to remove this as an error source. Figure 9.1 shows how earthrate modeling is used in strapdown inertial navigation. The implementation uses position information to calculate earthrate and remove it from the sensed rotation rates so that the coordinate transformation remains earth-fixed. Latitude and longitude rates due to horizontal velocities are also used to keep the navigation coordinates locally level.

The model for the sensible rotation rate vector $\vec{\Omega}_{\odot}$ in NED coordinates due to the rotation of the earth on its axis is

$$\vec{\Omega}_{\odot} = 7.29211510 \times 10^{-5} \begin{bmatrix} \cos(\lambda) \\ 0 \\ -\sin(\lambda) \end{bmatrix}, \tag{9.11}$$

where $7.29211510 \times 10^{-5}$ is the rotation rate of the earth in radians per second, and λ is the vehicle latitude.

[2]Circular error probable (CEP) is the radius of a horizontal circle at the estimated horizontal location such that it is equally likely that the actual location is inside or outside the circle. The CEP rate is the rate at which this radius grows with time.

9.3.5 Inertial Navigation Errors

Inertial navigators estimate the position, velocity, and attitude of their inertial sensor assemblies. INS navigation errors are the differences between the estimated position, velocity, and attitude and the true position, velocity, and attitude. These errors have three components of position, three components of velocity, and three components of attitude—a total of nine INS error state variables.

9.3.5.1 Sources of Navigation Errors INS navigation error dynamics are mostly due to

1. *Inertial sensor errors*, which introduce errors in attitude rate and acceleration. Sensor errors during INS alignment, when the INS is essentially stationary, contribute mostly to initial navigation errors in attitude. Thereafter, during navigation, accelerometer errors are integrated once to create velocity errors, which are integrated again to form position errors. Gyroscope errors are integrated once to create attitude errors.
2. *Navigating in earth-fixed coordinates*, which requires computed corrections for
   (a) *Accelerations due to gravity*, which cannot be sensed. This, by itself, causes instability of altitude error dynamics. Another dominant effect is that attitude errors in tilt (rotation about horizontal axes) cause a mismatch between the modeled gravitational acceleration and the countering sensed acceleration.
   (b) *Rotation of the earth-fixed coordinate frame*. In order for the INS to know which sensed rotation rates are due to the rotation of the earth, it must know its attitude with respect to the rotating earth. To maintain its orientation with respect to earth-fixed coordinates, the INS must keep track of the earth rotation axis direction. Miscalculation of earthrate correction due to attitude errors contributes to the growth of attitude errors.
   (c) *Coriolis effects*, miscalculation of which also contributes to navigation errors.

   All these effects couple together the dynamics of the different INS navigation errors in one linear stochastic system model.

9.3.5.2 Navigation Error State Variables We will derive a dynamic model for the dominant[3] dynamic characteristics of INS errors, using the NED coordinates shown in Figure 9.2.

[3]More detailed error model derivations are given in [283] (for gimbaled inertial navigators) and [235] (for strapdown navigators).

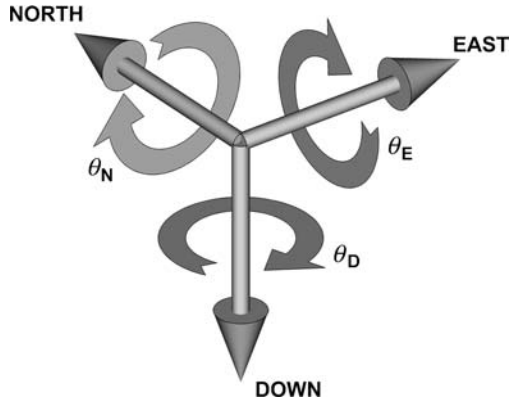


Figure 9.2 Locally-level NED coordinates.

The navigation error state vector

$$\varepsilon_{\text{INSNAV}} = \begin{bmatrix} \varepsilon_N \\ \varepsilon_E \\ \varepsilon_D \\ \dot{\varepsilon}_N \\ \dot{\varepsilon}_E \\ \dot{\varepsilon}_D \\ \varepsilon_{\theta N} \\ \varepsilon_{\theta E} \\ \varepsilon_{\theta D} \end{bmatrix}, \tag{9.12}$$

where the nine navigation error state variables in this coordinate system are

ε_N, the north component of position error.
ε_E, the east component of position error.
ε_D, the downward component of position error.
$\dot{\varepsilon}_N$, the north component of velocity error.
$\dot{\varepsilon}_E$, the east component of velocity error.
$\dot{\varepsilon}_D$, the downward component of velocity error.
$\varepsilon_{\theta N}$, the tilt (attitude error) about the north axis.
$\varepsilon_{\theta E}$, the tilt about the east axis.
$\varepsilon_{\theta D}$, the heading error, measured clockwise from north.

9.3.5.3 Vertical Error Instability Newton's universal law of gravitational acceleration is an inverse-square law. For a vehicle at a height h above the surface of the

earth with radius $R_\odot$, the calculated downward acceleration is

$$g = \frac{GM_\odot}{(R_\odot + h)^2}, \tag{9.13}$$

where G is the universal gravitational constant and $M_\odot$ is the mass of the earth.

If the estimated height is in error by an amount ε_D (measured downward), then the error in calculated gravitational acceleration will be $\ddot{\varepsilon}_D$, where

$$g + \ddot{\varepsilon}_D = \frac{GM_\odot}{(R_\odot + h - \varepsilon_D)^2} \tag{9.14}$$

$$\ddot{\varepsilon}_D \approx \left[-\left.\frac{\partial g}{\partial h}\right|_{\varepsilon_D = 0} \right] \varepsilon_D \tag{9.15}$$

$$\approx 2\frac{g}{R_\odot + h}\varepsilon_D \tag{9.16}$$

$$\approx 2\frac{g}{R_\odot}\varepsilon_D \text{ for } |h| \ll R_\odot \tag{9.17}$$

$$\approx \frac{2 \times 9.8[\text{m/s}^2]}{6372795.5 \text{ [m]}}\,\varepsilon_D \tag{9.18}$$

$$\approx \frac{1}{(520 \text{ [s]})^2}\,\varepsilon_D \tag{9.19}$$

$$\approx \tau_D^{-2}\,\varepsilon_D, \tag{9.20}$$

the solution to which is exponentially unstable. For $t \geq t_0$,

$$\varepsilon_D(t) = \varepsilon_D(t_0)\,\exp\left(\frac{t - t_0}{\tau_D}\right) \tag{9.21}$$

$$\tau_D \approx \sqrt{\frac{R_\odot}{2 \times g}} \tag{9.22}$$

$$\approx 520 \text{ [s]}. \tag{9.23}$$

Although INS errors in the vertical direction may be unstable, it is not a problem for ships at sea. Their altitudes are essentially fixed near sea level. Otherwise, some form of auxiliary sensor aiding is required to control vertical INS errors. Altimeter inputs have been used in inertial navigation of aircraft, which usually carry altimeters anyway. GNSS can also be used for this purpose.

9.3.5.4 Schuler Constants A factor of τ_D in Equation 9.22 is the related time constant

$$\tau_S = \sqrt{2}\,\tau_D \tag{9.24}$$

$$= \sqrt{\frac{R_\odot}{g}} \tag{9.25}$$

$$\approx 806.4 \text{ s} \tag{9.26}$$

$$\approx \frac{84.4 \text{ min}}{2\pi \text{ rad}} \tag{9.27}$$

$$\stackrel{\text{def}}{=} 1/\Omega_S. \tag{9.28}$$

τ_S is called the *Schuler time-constant*, 84.4 min is called the *Schuler period*, and $\Omega_S \approx 0.0012$ radian/s ≈ 0.0002 Hz is called the *Schuler frequency*—all named after Maximilian Schuler (1882–1972) for his 1923 paper [242] about the errors of gyrocompasses, which were then replacing magnetic compasses on iron ships.

Schuler constants are also fundamental to the dynamics of INS navigation errors.

9.3.5.5 Tilts Two of the three navigation error state variables for attitude errors are called *tilts*. The terminology was established early in the development of gimbaled systems, when inertial sensor assemblies were oriented with their sensor axes parallel to the north, east, and vertical directions, as illustrated in Figures 9.3 and 9.4(*a*).

Tilt errors then referred to physical tilts of the ISA, as illustrated in Figure 9.4. With gimbaled inertial *platforms*, these tilts could be measured in the laboratory using optical autocollimators and mirrors mounted on the ISA.

There may be some ambiguity about whether *north tilt* means "tilting toward the north" or "tilting about the north axis," but the latter definition illustrated in Figures 9.3 and 9.4 is more common in practice.

Tilts as Rotations About Horizontal Axes *North tilt* refers to small-angle rotations around the north-pointing axis, as shown in Figure 9.4(*b*).

East tilt refers to small-angle rotations around the east-pointing axis, as illustrated in Figure 9.4(*c*), and rotations about the downward vertical axis are called *heading errors*, as illustrated in Figure 9.4(*d*).

North tilts cause a nominally east-pointing sensor input axis to point a little downward from true level, as illustrated in Figure 9.5.

When the north tilt $\varepsilon_{\theta N} > 0$, the INS estimate of the vertical direction is rotated toward the east, and the (unsensed) estimated gravitational acceleration $\hat{g}$ will have a westerly component, as illustrated in Figure 9.3, leading to a negative net east acceleration error in the INS implementation.

Similarly, east tilt means that a nominally north-pointing sensor input axis is actually pointing a little upward from true level.

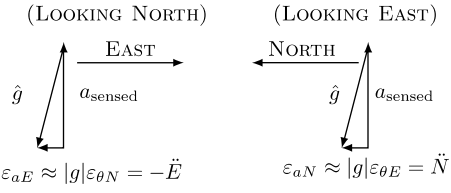


Figure 9.3 Dynamic effects of tilt errors $\varepsilon_{\theta N}$ and $\varepsilon_{\theta E}$.

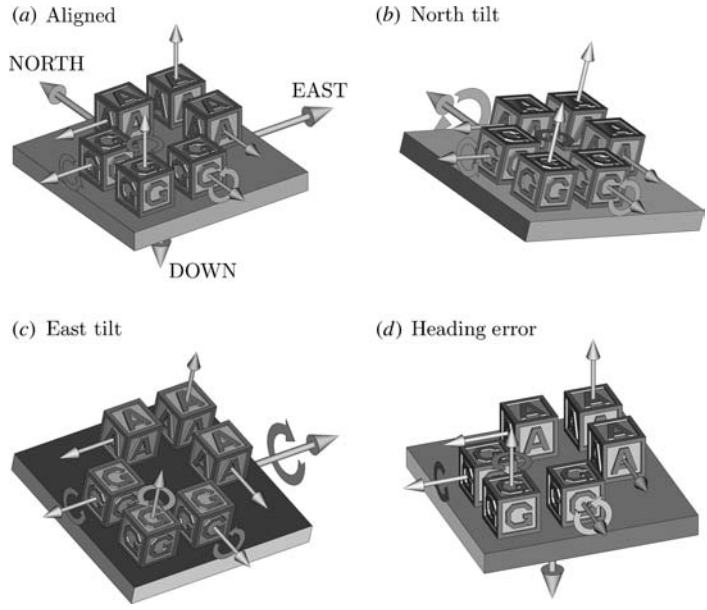


Figure 9.4 Conventions for tilts and heading errors (gimbaled ISA).

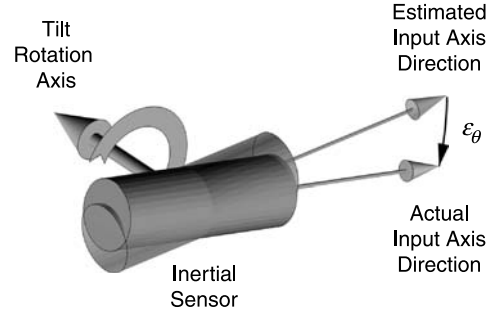


Figure 9.5 North tilt of a nominally east-pointing accelerometer.

For strapdown systems, *tilt* refers to an ISA attitude rotational error about the north or east axis.

Dynamic Effects of Tilt Errors Figure 9.6 illustrates how attitude errors and position errors cause acceleration errors due to miscalculation of gravity relative to the sensed acceleration.

Figure 9.6 shows how an INS at rest with tilt error $\varepsilon_{\theta E}$ about the east axis creates an acceleration error $\ddot{\varepsilon}_N$ in the northward direction. Due to the tilt, the INS computes gravitational acceleration g that is not in the direction of the sensed

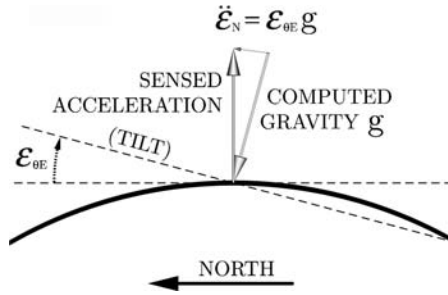


Figure 9.6 Acceleration errors due to tilt position errors.

acceleration. When the two are added vectorially, the sum is not zero. The net acceleration error

$$\ddot{\varepsilon}_N \approx \varepsilon_{\theta E} \times g \tag{9.29}$$

$$g \approx 9.8 \ [\mathrm{m/s^2}]. \tag{9.30}$$

9.3.5.6 Schuler Oscillations Latitude rate is calculated in an INS as

$$\dot{\lambda} = \frac{V_N}{R_\odot + h}, \tag{9.31}$$

where λ is INS latitude, V_N is north velocity of the INS, h is the altitude of the INS, and $R_\odot + h$ is the radius of curvature of the gravitational equipotential surface.

If $\dot{\varepsilon}_N$ is the INS north velocity error, it causes a tilt rate $\dot{\varepsilon}_{\theta E}$ about the east axis such that

$$\dot{\lambda} - \dot{\varepsilon}_{\theta E} = \frac{V_N + \dot{\varepsilon}_N}{R_\odot + h} \tag{9.32}$$

$$\dot{\varepsilon}_{\theta E} = -\frac{\dot{\varepsilon}_N}{R_\odot + h} \tag{9.33}$$

$$\dot{\varepsilon}_N = -(R_\odot + h)\dot{\varepsilon}_{\theta E} \tag{9.34}$$

$$\approx -R_\odot \dot{\varepsilon}_{\theta E}. \tag{9.35}$$

However, according to Equation 9.29,

$$\dot{\varepsilon}_{\theta E} \approx \frac{d}{dt}\dot{\varepsilon}_N / g, \tag{9.36}$$

from which

$$\frac{d^2}{dt^2}\dot{\varepsilon}_N \approx -\left(\frac{g}{R_\odot}\right)\dot{\varepsilon}_N \tag{9.37}$$

$$\approx -\Omega_S^2\dot{\varepsilon}_N, \tag{9.38}$$

where the Schuler frequency Ω_S is given in Equation 9.28.

Equation 9.37 is the second-order differential equation for an undamped harmonic oscillator with resonant frequency Ω_S. This oscillatory behavior of INS horizontal velocity errors is called *Schuler oscillation*. It has a period of 84.4 min, the Schuler period.

Schuler oscillations also occur in the east tilt error rate $\dot{\varepsilon}_{\theta E}$, east velocity error rate $\dot{\varepsilon}_E$, and north tilt error rate $\dot{\varepsilon}_{\theta N}$.

Figure 9.7 is a plot of Schuler oscillations generated by the MATLAB script `SchulerDemo.m`. It shows that, due to Coriolis effects, Schuler oscillations make INS horizontal errors behave like a Foucault pendulum. This is the classic pattern of behavior for INS errors. It is standard practice during INS development testing to induce horizontal velocity errors in order to verify INS software implementation by observing the resulting Schuler oscillation pattern.

Schuler oscillations also occur in horizontal position errors and tilt errors, but they may oscillate about a nonzero value. For example, the conditions shown in Figure 9.8 represent a first-order equilibrium state in which the effects of tilt errors and position errors cancel one another. The computed gravitational acceleration exactly cancels the sensed acceleration when $\varepsilon_{\theta N} = \varepsilon_E/R_\odot$. Introducing a horizontal

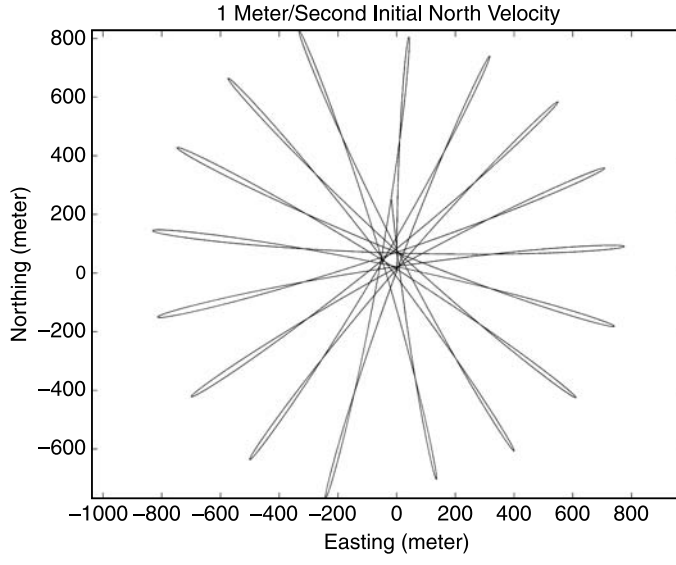


Figure 9.7 Schuler oscillations with no offset.

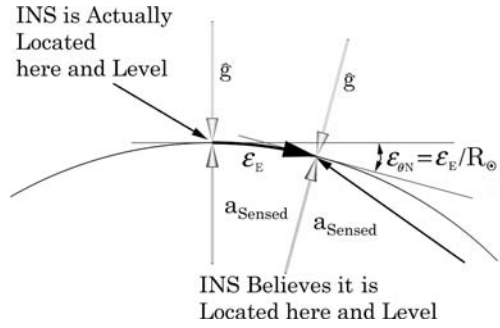


Figure 9.8 Position and tilt errors at equilibrium.

velocity error in that position would cause the tilt errors and horizontal position errors to oscillate about this nonzero equilibrium state. Figure 9.9 shows such an example as the result of an initial east tilt error of 0.1 milliradian, which results in a Schuler oscillation around an offset north position error of about 600 m.

9.3.5.7 Navigation Error Dynamics The linear model for navigation error dynamics will have the general form

$$\frac{d}{dt}\begin{bmatrix} \varepsilon_{INSNAV} \\ \varepsilon_{\text{INSENS}} \end{bmatrix} = \begin{bmatrix} F_{NN} & F_{NS} \\ 0 & F_{SS} \end{bmatrix}\begin{bmatrix} \varepsilon_{INSNAV} \\ \varepsilon_{\text{INSENS}} \end{bmatrix} + \begin{bmatrix} 0 \\ w_{\text{INSENS}}(t) \end{bmatrix}, \tag{9.39}$$

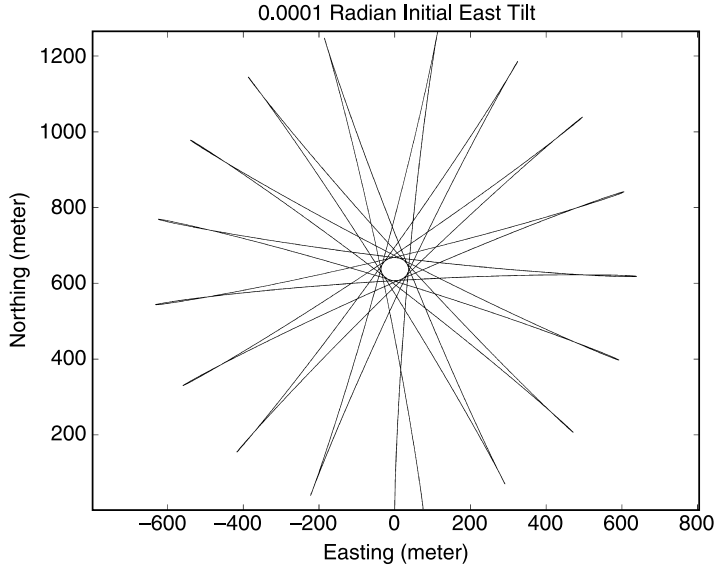


Figure 9.9 Schuler oscillations with offset.

where $\varepsilon_{\text{INSNAV}}$ is the navigation error state vector given by Equation 9.12 and $\varepsilon_{\text{INSENS}}$ is the sensor error state vector, which is yet to be defined. Here, we will define the 3×3 blocks of the submatrix

$$F_{NN} = \begin{bmatrix} F_{pv} & I_3 & F_{p\theta} \\ F_{vp} & F_{vv} & F_{v\theta} \\ F_{\theta p} & F_{\theta v} & F_{\theta\theta} \end{bmatrix}. \tag{9.40}$$

The state vector $\varepsilon_{\text{INSENS}}$ and submatrices F_{NS} and F_{SS} will be defined in Section 9.3.6.

Error Dynamics Due to Attitude Errors The principal effects of attitude errors on navigation error dynamics are due to miscalculations of the direction of gravity and the direction of the earth rotation axis. These effects can be modeled in matrix form by using the first-order small-angle rotation error matrix

$$[\varepsilon_\theta] = \begin{bmatrix} 0 & -\varepsilon_{\theta D} & \varepsilon_{\theta E} \\ \varepsilon_{\theta D} & 0 & -\varepsilon_{\theta N} \\ -\varepsilon_{\theta E} & \varepsilon_{\theta N} & 0 \end{bmatrix} \tag{9.41}$$

to represent errors in resolving accelerations and attitude rates into navigation coordinates.

Gravity Miscalculation The first-order contribution of attitude errors to acceleration errors can be modeled as

$$\frac{d}{dt}\begin{bmatrix} \dot{\varepsilon}_N \\ \dot{\varepsilon}_E \\ \dot{\varepsilon}_D \end{bmatrix} \approx \begin{bmatrix} 0 & -\varepsilon_{\theta D} & \varepsilon_{\theta E} \\ \varepsilon_{\theta D} & 0 & -\varepsilon_{\theta N} \\ -\varepsilon_{\theta E} & \varepsilon_{\theta N} & 0 \end{bmatrix}\begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \tag{9.42}$$

$$\approx \begin{bmatrix} g\varepsilon_{\theta E} \\ -g\varepsilon_{\theta N} \\ 0 \end{bmatrix} \tag{9.43}$$

$$= \begin{bmatrix} 0 & g & 0 \\ -g & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} \varepsilon_{\theta N} \\ \varepsilon_{\theta E} \\ \varepsilon_{\theta D} \end{bmatrix}. \tag{9.44}$$

Resolving Other Accelerations Attitude errors cause additional acceleration errors when sensed accelerations are resolved into navigation coordinates. We have already addressed the effects due to gravity and its countering sensed acceleration $a_D = -g$. The other error mechanisms can be modeled as another contribution from attitude errors to sensed acceleration

$$a_{\text{SENSED}} = \begin{bmatrix} a_N \\ a_E \\ a_D \end{bmatrix} \tag{9.45}$$

as

$$\frac{d}{dt}\begin{bmatrix}\dot{\varepsilon}_N\\ \dot{\varepsilon}_E\\ \dot{\varepsilon}_D\end{bmatrix} \approx \begin{bmatrix}0 & -\varepsilon_{\theta D} & \varepsilon_{\theta E}\\ \varepsilon_{\theta D} & 0 & -\varepsilon_{\theta N}\\ -\varepsilon_{\theta E} & \varepsilon_{\theta N} & 0\end{bmatrix}\begin{bmatrix}a_N\\ a_E\\ a_D+g\end{bmatrix} \tag{9.46}$$

$$\approx \begin{bmatrix}0 & a_D+g & -a_N\\ -a_D-g & 0 & a_E\\ a_N & -a_E & 0\end{bmatrix}\begin{bmatrix}\varepsilon_{\theta N}\\ \varepsilon_{\theta E}\\ \varepsilon_{\theta D}\end{bmatrix}. \tag{9.47}$$

The combined model for the effect of attitude errors on acceleration errors will then be

$$\frac{d}{dt}\begin{bmatrix}\dot{\varepsilon}_N\\ \dot{\varepsilon}_E\\ \dot{\varepsilon}_D\end{bmatrix} \approx \underbrace{\begin{bmatrix}0 & a_D+2g & -a_N\\ -a_D-2g & 0 & a_E\\ a_N & -a_E & 0\end{bmatrix}}_{F_{v\theta}}\begin{bmatrix}\varepsilon_{\theta N}\\ \varepsilon_{\theta E}\\ \varepsilon_{\theta D}\end{bmatrix}, \tag{9.48}$$

where, under nominal conditions with the INS at rest,

$$a_{\text{SENSED}} = \begin{bmatrix}0\\ 0\\ -g\end{bmatrix}, \tag{9.49}$$

and Equations 9.44 and 9.48 are identical.

Earthrate Miscalculation In similar fashion, the first-order contribution of attitude errors to attitude rate errors will be

$$\frac{d}{dt}\begin{bmatrix}\varepsilon_{\theta N}\\ \varepsilon_{\theta E}\\ \varepsilon_{\theta D}\end{bmatrix} \approx \begin{bmatrix}0 & -\varepsilon_{\theta D} & \varepsilon_{\theta E}\\ \varepsilon_{\theta D} & 0 & -\varepsilon_{\theta N}\\ -\varepsilon_{\theta E} & \varepsilon_{\theta N} & 0\end{bmatrix}\begin{bmatrix}\Omega_\odot\cos(\lambda)\\ 0\\ -\Omega_\odot\sin(\lambda)\end{bmatrix} \tag{9.50}$$

$$\approx \Omega_\odot\begin{bmatrix}-\varepsilon_{\theta E}\sin(\lambda)\\ \varepsilon_{\theta D}\cos(\lambda)+\varepsilon_{\theta N}\sin(\lambda)\\ -\varepsilon_{\theta E}\cos(\lambda)\end{bmatrix} \tag{9.51}$$

$$\approx \underbrace{\begin{bmatrix}0 & -\Omega_\odot\sin(\lambda) & 0\\ \Omega_\odot\sin(\lambda) & 0 & \Omega_\odot\cos(\lambda)\\ 0 & -\Omega_\odot\cos(\lambda) & 0\end{bmatrix}}_{F_{\dot{\theta}\theta}}\begin{bmatrix}\varepsilon_{\theta N}\\ \varepsilon_{\theta E}\\ \varepsilon_{\theta D}\end{bmatrix}, \tag{9.52}$$

where λ is the latitude of the INS.

Error Dynamics Due to Position Errors From Equation 9.20, the contribution of position errors to velocity errors will be

$$\frac{d}{dt}\begin{bmatrix}\dot{\varepsilon}_N\\ \dot{\varepsilon}_E\\ \dot{\varepsilon}_D\end{bmatrix} \approx \underbrace{\begin{bmatrix}-\tau_S^{-2} & 0 & 0\\ 0 & -\tau_S^{-2} & 0\\ 0 & 0 & \tau_D^{-2}\end{bmatrix}}_{F_{vp}}\begin{bmatrix}\varepsilon_N\\ \varepsilon_E\\ \varepsilon_D\end{bmatrix} \tag{9.53}$$

$$\tau_S \approx 806.4 \text{ s} \tag{9.54}$$

$$\tau_D \approx 520 \text{ s}. \tag{9.55}$$

Coriolis Effect The apparent acceleration due to the Coriolis effect in rotating coordinates is

$$a_{\text{Coriolis}} = -2\,\Omega \otimes V, \tag{9.56}$$

where a_{Coriolis} is the Coriolis acceleration, V is velocity in rotating coordinates, $\otimes$ is the vector cross-product, and the coordinate rotation rate vector is Ω. In earth-fixed NED coordinates, the coordinate rotation rate vector

$$\Omega = \Omega_\odot \begin{bmatrix}\cos(\lambda)\\ 0\\ -\sin(\lambda)\end{bmatrix} \tag{9.57}$$

$$\Omega_\odot \approx 7.292115 \times 10^{-5} \text{ [rad/s]} \tag{9.58}$$

$$\lambda \stackrel{\text{def}}{=} \text{latitude of INS location.}$$

Therefore, the contribution of Coriolis acceleration due to velocity errors will be

$$\frac{d}{dt}\begin{bmatrix}\dot{\varepsilon}_N\\ \dot{\varepsilon}_E\\ \dot{\varepsilon}_D\end{bmatrix} = \underbrace{\begin{bmatrix}0 & -2\,\Omega_\odot \sin(\lambda) & 0\\ 2\,\Omega_\odot \sin(\lambda) & 0 & 2\,\Omega_\odot \cos(\lambda)\\ 0 & -2\,\Omega_\odot \cos(\lambda) & 0\end{bmatrix}}_{F_{vv}}\begin{bmatrix}\dot{\varepsilon}_N\\ \dot{\varepsilon}_E\\ \dot{\varepsilon}_D\end{bmatrix}. \tag{9.59}$$

Effect of Velocity Errors on Tilt Rates Horizontal velocities are used in calculating latitude and longitude rates, and velocity errors cause tilt rate errors:

$$\frac{d}{dt}\begin{bmatrix}\varepsilon_{\theta N}\\ \varepsilon_{\theta E}\\ \varepsilon_{\theta D}\end{bmatrix} = \underbrace{\begin{bmatrix}0 & R_\odot^{-1} & 0\\ -R_\odot^{-1} & 0 & 0\\ 0 & 0 & 0\end{bmatrix}}_{F_{\theta v}}\begin{bmatrix}\dot{\varepsilon}_N\\ \dot{\varepsilon}_E\\ \dot{\varepsilon}_D\end{bmatrix}. \tag{9.60}$$

9.3.5.8 Dynamic Coefficient Matrix The submatrices $F_{v\theta}$, $F_{\theta\theta}$, F_{vp}, F_{vv}, and $F_{\theta v}$ of the dynamic coefficient matrix F_{NN} of Equation 9.40 are derived above in Section 9.3.5.7. The remaining 3×3 blocks of F_{NN} are zero, leaving the following first-order value for the 9×9 dynamic coefficient matrix for INS navigation errors:

$$F_{NN} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -2\Omega_{\odot} s_{\lambda} & 0 & 0 & a_D + 2g & -a_N \\ 0 & 0 & 0 & 2\Omega_{\odot} s_{\lambda} & 0 & 2\Omega_{\odot} c_{\lambda} & -a_D - 2g & 0 & a_E \\ 0 & 0 & \tau_D^{-2} & 0 & -2\Omega_{\odot} c_{\lambda} & 0 & a_N & -a_E & 0 \\ 0 & 0 & 0 & 0 & R_{\odot}^{-1} & 0 & 0 & -\Omega_{\odot} s_{\lambda} & 0 \\ 0 & 0 & 0 & -R_{\odot}^{-1} & 0 & 0 & \Omega_{\odot} s_{\lambda} & 0 & \Omega_{\odot} c_{\lambda} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\Omega_{\odot} c_{\lambda} & 0 \end{bmatrix} \tag{9.61}$$

$$\left.\begin{aligned} \tau_S &\approx 806.4 \text{ s} \\ \tau_D &\approx 520 \text{ s} \\ g &\approx 9.8 \text{ [m/s}^2\text{]} \\ R_{\odot} &\approx 6372795.5 \text{ [m]} \\ \Omega_{\odot} &\approx 7.292115 \times 10^{-5} \text{ [rad/s]} \\ s_{\lambda} &\stackrel{\text{def}}{=} \sin(\lambda) \\ c_{\lambda} &\stackrel{\text{def}}{=} \cos(\lambda) \\ \lambda &\stackrel{\text{def}}{=} \text{INS latitude} \\ a_N &\stackrel{\text{def}}{=} \text{north INS acceleration [m/s}^2\text{]} \\ a_E &\stackrel{\text{def}}{=} \text{east INS acceleration [m/s}^2\text{]} \\ a_D &\stackrel{\text{def}}{=} \text{downward INS acceleration [m/s}^2\text{]} \end{aligned}\right\} \tag{9.62}$$

The dynamic coefficient matrix F_{NN} is computed by the MATLAB function `FNNmat.m` on the CD.

Example 9.1 (Barometer Stabilization of INS Altitude) This example uses a barometric altimeter to stabilize the altitude of an INS. The barometer is assumed to have an RMS altitude error of 100 m with a correlation time-constant of 8 h, plus RMS white noise of 10 m. The INS is assumed to be driven by RMS process noise of 1 μg per sample interval of 1 s and attitude process noise of 0.01 deg/h RMS.

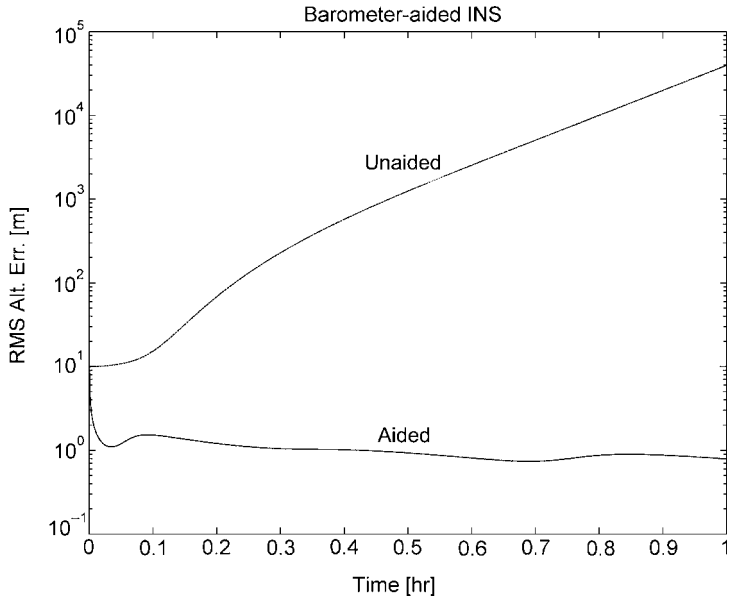


Figure 9.10 Barometer-aided INS altitude uncertainties versus time.

Figure 9.10 shows the resulting RMS altitude uncertainty over 1 h, with and without barometer aiding. The aided altitude uncertainty settles at around 1 m, whereas the RMS uncertainty for the unaided case climbs to about 40 km in 1 h.

This plot was generated by the MATLAB m-file `Example9pt1.m` on the CD.

9.3.6 Inertial Sensor Errors and Noise

Inertial navigators integrate the outputs of sensors, which makes them relatively insensitive to additive zero-mean white sensor noise. However, they are very sensitive to output errors that are not zero-mean over periods of minutes or hours.

Example 9.2 (Simplified Sensitivity to Accelerometer Errors) A very small constant accelerometer output error

$$\varepsilon_a = 10^{-5} g = 9.8 \times 10^{-5} \text{ [m/s]}^2,$$

doubly integrated for 2 h (7200 s, or about the time it takes to fly from Los Angeles to Seattle) produces a position error of

$$\varepsilon_p = \frac{1}{2}\varepsilon_a (7200)^2 \tag{9.63}$$

$$\approx \frac{1}{2} \times 9.8 \times 10^{-5} \times 5.2 \times 10^7 \text{ [m]} \tag{9.64}$$

$$\approx 2.5 \text{ km}, \tag{9.65}$$

which is more than two orders of magnitude larger than RMS navigation errors from satellite navigation.

Example 9.3 (10^{-5} g Accelerometer Bias) We can simulate the conditions of Example 9.2 (10^{-5} g accelerometer bias error) using the MATLAB function `FSSmat.m` to see how the simple integration model of Example 9.2 compares with a more faithful model of INS errors.

Figure 9.11 is a set of plots generated by simulating a north accelerometer bias error of 10^{-5} g for 1 h. It shows about one cycle of Schuler oscillation in north position error with a peak-to-peak variation on the order of 70 m, plus some Coriolis coupling into the east and vertical directions. (Errors in the vertical direction are unstable.)

The peak error is far less than predicted by the simple model of Example 9.2 but still bigger than RMS GNSS navigation errors.

This plot was generated by the MATLAB m-file `Example9pt3.m`, which is on the CD. This script simulates the full nine-state INS error state.

Because small sensor errors can dominate INS performance, it has become standard practice to use sensor error models in the design, development, evaluation, and operation of INS and in the integration of an INS with other navigation methods. These models are generally specialized to the set of sensors used in a particular implementation, and they can have more than 100 calibration parameters to be estimated [11].

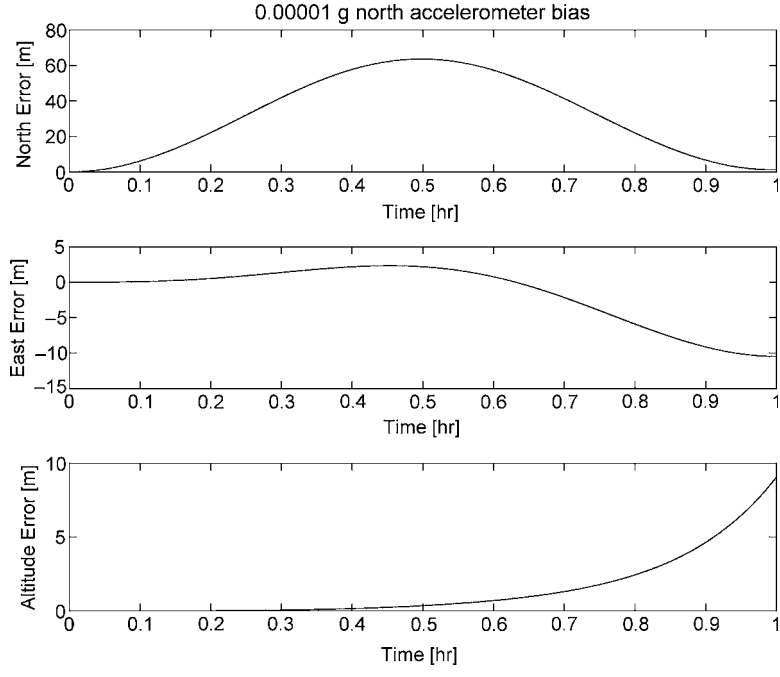


Figure 9.11 Simulated INS errors from 10^{-5} g north accelerometer bias.

The sensor error model derived here is relatively simple and generic in nature, with 21 state variables.

9.3.6.1 Inertial Sensor Requirements

Inertial-Grade Sensors The most demanding accuracy requirements for inertial sensors are for what are called *inertial-grade* instruments. These are generally for *free inertial* INS navigation (i.e., without using an auxiliary sensor except for altitude stabilization).

Gyroscope Requirements In order to sense the rotation of the earth at ≈ 15 deg/h (≈ 15 arc-s/s, or $\approx 7.3 \times 10^{-5}$ radian/s) during alignment, gyroscope errors should be at least a few orders of magnitude smaller. Medium-accuracy inertial navigation (1 nautical mile per hour CEP rate) requires ≈ 0.01 deg/h or less RMS gyroscope drift rates.

Accelerometer Requirements Inertial-grade accelerometers have demanding accuracy requirements for operation at input frequencies from dc to about 10^2 or 10^3 Hz. Accelerometer errors for medium-accuracy inertial navigation are generally on the order of 10^{-6} g or better.

Calibration Requirements It is not practical to achieve these levels of sensor accuracy by precise manufacturing. This is usually done after manufacture, by calibrating the sensor errors (i.e., fitting them to phenomenological models) and using the resulting fitted parameter values with the same models to compensate the sensor outputs during operation. Manufacturing is then tasked to make these errors as stable as practical to avoid frequent recalibration.

Compensation Model For many practical applications, it suffices to use a scale factor and bias (output offset) for each sensor axis and perhaps two components of input axis misalignment angles. For a complete INS sensor cluster with three gyroscopes and three accelerometers, the resulting sensor compensation models will have the form

$$z_{\text{input}} = M(z_{\text{output}} - z_{\text{bias}}), \tag{9.66}$$

where z_{bias} is the output bias (either acceleration or attitude rate) and M is a 3×3 matrix of scale factors and input axis misalignments. This implementation requires a total of 12 compensation parameters for the accelerometers and another 12 for the gyroscopes.

Even for relatively low-cost INS designs, the input axis misalignments tend to be rather stable compared to the biases and scale factors. In that case, the information gained from integration with GNSS can be used to track time-varying values of the gyroscope and accelerometer biases and scale factors. The total number of sensor calibration parameters in this case is six for the gyroscopes and another six for the accelerometers.

Sensor Cost Issues Calibration takes time, labor, and expensive facilities and equipment, which add significantly to sensor cost. The set of parameters required to

achieve this level of accuracy can number in the dozens or even hundreds, depending on INS system design and performance requirements. (See, e.g., [106, 166] for more details.)

9.3.6.2 Applications of Sensor Error Models Inertial sensor error modeling is used in a number of ways in inertial navigation technology:

1. Phenomenological models are used in the design of accelerometers and gyroscopes to predict and control their dynamic ranges, accuracy levels, temperature sensitivities, and other error mechanisms.
2. Manufacturers test their sensors and publish models of their error characteristics so that INS designers can select sensors to meet their own system requirements. System-level performance analysis uses the Riccati equation of the Kalman filter for predicting performance.
3. These same sensor models are also used in Kalman filters for calibration and compensation of sensor errors in the INS, and in the associated Riccati equation for performance analysis of integrated GNSS/INS navigation systems.
4. The software implementation for integrated GNSS/INS navigation can use the same models in Kalman filters for recalibrating and compensating for sensor errors during navigation.

9.3.6.3 Sensor-Fixed XYZ Coordinates Because inertial sensors are rigidly aligned within an ISA, sensor errors are usually modeled in an ISA-fixed coordinate system. It simplifies the math if we let this be an *XYZ* coordinate system with its *X*, *Y*, and *Z* axes parallel to the nominal input axes of a triad of sensors, as illustrated in Figure 9.12. This shows a triad of three accelerometers with orthogonal input axes,

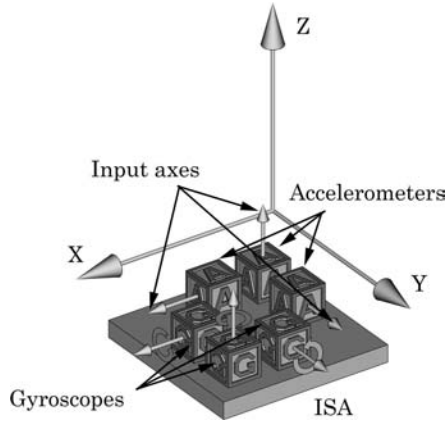


Figure 9.12 ISA-fixed *XYZ* coordinates.

aligned with the input axes of a triad of gyroscopes. The *XYZ* coordinate axes are aligned with the orthogonal sensor input axes.

9.3.6.4 First-Order Sensor Error Models First-order sensor errors include biases (one per sensor), scale factor variations (one per sensor), and input axis misalignments (two per sensor).

The first-order model for inertial sensor errors has the form[4]

$$y_{\text{output}} = Ly_{\text{input}} + b_y \tag{9.67}$$

$$L = \begin{bmatrix} S_{XX} & M_{XY} & M_{XZ} \\ M_{YX} & S_{YY} & M_{YZ} \\ M_{ZX} & M_{ZY} & S_{ZZ} \end{bmatrix}, \tag{9.68}$$

This is essentially the *compensation model* of Equation 9.66, with $L = M^{-1}$. In Equation 9.66, we needed to compensate for sensor errors by representing the input as a function of the output. In this case, we need to represent the errors in sensor outputs due to shifts in the compensation parameters.

Equation 9.68 assumes the ISA-fixed *XYZ* coordinate system of Section 9.3.6.3, with axes along the nominal input axes of three orthogonal inertial sensors. For example, the *X*-accelerometer and *X*-gyroscope input axes would nominally be along the *X*-axis direction.

In Equation 9.67, y represents either accelerations (for accelerometers) or rotation rates (for gyroscopes), y_{input} is the vector of input values, y_{output} is the vector of input values, and b_y is the vector of sensor biases. The matrix L in Equation 9.68 is called a *scale factor and misalignment matrix*. The diagonal terms S_{ii} of L represent sensor scale factors and the off-diagonal terms M_{ij} represent sensor input axis misalignments. The elements of L and b_y are the model parameters.

Integrated GNSS/INS operation is capable of recalibrating the INS to compensate for small variations in the sensor compensation parameters during operation. As a rule, this recalibration is only looking for small variations from the laboratory-determined values, including variations in

accelerometer bias errors ε_{abX}, ε_{abY}, and ε_{abZ}.

accelerometer scale factor errors ε_{asX}, ε_{asY}, and ε_{asZ}.

accelerometer input axis misalignment errors ε_{aXY}, ε_{aXZ} and ε_{aYX}, ε_{aYZ}, ε_{aZX}, and ε_{aZY}.

gyroscope bias errors ε_{gbX}, ε_{gbY}, and ε_{gbZ}.

gyroscope scale factor errors ε_{gsX}, ε_{gsY}, and ε_{gsZ}.

gyroscope input axis misalignment errors ε_{gXY}, ε_{gXZ} and ε_{gYX}, ε_{aYZ}, ε_{aZX}, and ε_{aZY}.

[4]A combination of displacement b_y and linear transformation L is called an *affine transformation*.

Unobservable Sensor Misalignment Variations In a nominally orthogonal triad of sensors (either gyroscopes or accelerometers), each sensor input axis direction has 2 degrees of freedom for misalignments—for a total of 6 degrees of freedom for a three-sensor subassembly (either gyroscopes or accelerometers) and 12 degrees of freedom for the six sensors in the ISA (three gyroscopes and three accelerometers). However, 3 of those 12 degrees of freedom correspond to rigid-body rotations of the ISA with respect to the host vehicle, which is part of the INS error model. That leaves only nine degrees of freedom in sensor input axis misalignments as part of the sensor error model.

For recalibration of b_y and L using GNSS/INS integration, the model needs to be trimmed to eliminate three unobservable misalignment parameters. This can be done using small-angle approximation by forcing the variation in the misalignment matrix ε_{La} (for the accelerometers) to be symmetric:

$$\varepsilon_{La} = \begin{bmatrix} \varepsilon_{aX} & \varepsilon_{aXY} & \varepsilon_{aZX} \\ \varepsilon_{aXY} & \varepsilon_{aY} & \varepsilon_{aYZ} \\ \varepsilon_{aZX} & \varepsilon_{aYZ} & \varepsilon_{aZ} \end{bmatrix}. \tag{9.69}$$

This, in effect, lets the accelerometer input axis directions define the ISA reference attitude, which is part of the INS error model. The misalignments of the gyroscope input axis directions with respect to the accelerometer input axes remain part of the sensor error model.

9.3.6.5 Sensor Error State Variables The structure of the dynamic coefficient matrices for these sensor error parameters is simplified somewhat if we model the symmetric and antisymmetric parts of L,

$$L_s = \frac{1}{2}\left[L + L^T\right] \tag{9.70}$$

$$L_a = \frac{1}{2}\left[L - L^T\right], \tag{9.71}$$

respectively. Equivalently, we can express the matrix L in terms of separate parameters for the symmetric and antisymmetric parts, as

$$L = \begin{bmatrix} \varepsilon_{\mathrm{sX}} & \varepsilon_{\mathrm{XYs}} - \varepsilon_{\mathrm{XYa}} & \varepsilon_{\mathrm{ZXs}} + \varepsilon_{\mathrm{ZXa}} \\ \varepsilon_{\mathrm{XYs}} + \varepsilon_{\mathrm{XYa}} & \varepsilon_{\mathrm{sY}} & \varepsilon_{\mathrm{YZs}} - \varepsilon_{\mathrm{YZa}} \\ \varepsilon_{\mathrm{ZXs}} - \varepsilon_{\mathrm{ZXa}} & \varepsilon_{\mathrm{YZs}} + \varepsilon_{\mathrm{YZa}} & \varepsilon_{\mathrm{sZ}} \end{bmatrix}, \tag{9.72}$$

where the final subscript "s" refers to the symmetric part and the corresponding final subscript "a" refers to the antisymmetric part. The resulting 21-dimensional sensor

error state vector will have the form

$$x = \begin{bmatrix} x_a \\ x_g \end{bmatrix}, \quad x_a = \begin{bmatrix} \varepsilon_{\text{abX}} \\ \varepsilon_{\text{abY}} \\ \varepsilon_{\text{abZ}} \\ \varepsilon_{\text{asX}} \\ \varepsilon_{\text{asY}} \\ \varepsilon_{\text{asZ}} \\ \varepsilon_{\text{aYZs}} \\ \varepsilon_{\text{aZXs}} \\ \varepsilon_{\text{aXYs}} \end{bmatrix}, \quad x_g = \begin{bmatrix} \varepsilon_{\text{gbX}} \\ \varepsilon_{\text{gbY}} \\ \varepsilon_{\text{gbZ}} \\ \varepsilon_{\text{gsX}} \\ \varepsilon_{\text{gsY}} \\ \varepsilon_{\text{gsZ}} \\ \varepsilon_{\text{gYZs}} \\ \varepsilon_{\text{gZXs}} \\ \varepsilon_{\text{gXYs}} \\ \varepsilon_{\text{gYZa}} \\ \varepsilon_{\text{gZXa}} \\ \varepsilon_{\text{gXYa}} \end{bmatrix}, \tag{9.73}$$

where there are three more sensor error state variables for the gyroscopes than for the accelerometers.

9.3.6.6 Sensor Error Dynamics Models for the sensor errors are usually random walk models or exponentially correlated process models. The latter are useful because they allow the analyst to use model parameters (i.e., RMS error and error correlation time) derived from test data.

The dynamic model for an independent, exponentially correlated process has the form

$$\frac{d}{dt}\varepsilon = \frac{-1}{\tau}\varepsilon + w(t), \tag{9.74}$$

with specified correlation time-constant τ and white process noise $w(t) \in \mathcal{N}(0, 2\,\sigma^2/\tau)$, where $\sigma^2 = \mathrm{E}\langle\varepsilon^2\rangle$ is the mean-squared sensor error.

For the 21 sensor error state variables listed in Section 9.3.6.4, Equation 9.74 would have the state-space form

$$\frac{d}{dt}x_S = F_{SS}x_S + w_S(t), \tag{9.75}$$

where

$$\begin{aligned} x_S^T = \big[& \varepsilon_{\text{abX}}, \quad \varepsilon_{\text{abY}}, \quad \varepsilon_{\text{abZ}}, \quad \varepsilon_{\text{asX}}, \quad \varepsilon_{\text{asY}}, \quad \varepsilon_{\text{asZ}}, \quad \varepsilon_{\text{aYZs}}, \quad \varepsilon_{\text{aZXs}}, \quad \varepsilon_{\text{aXYs}}, \\ & \varepsilon_{\text{gbX}}, \quad \varepsilon_{\text{gbY}}, \quad \varepsilon_{\text{gbZ}}, \quad \varepsilon_{\text{gsX}}, \quad \varepsilon_{\text{gsY}}, \quad \varepsilon_{\text{gsZ}}, \quad \varepsilon_{\text{gYZs}}, \quad \varepsilon_{\text{gZXs}}, \quad \varepsilon_{\text{gXYs}}, \\ & \varepsilon_{\text{gYZa}}, \quad \varepsilon_{\text{gZXa}}, \quad \varepsilon_{\text{gXYa}} \big] \end{aligned} \tag{9.76}$$

$$F_{SS}=\begin{bmatrix} -\tau_{abX}^{-1} & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & -\tau_{abY}^{-1} & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & -\tau_{abZ}^{-1} & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & -\tau_{asX}^{-1} & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\tau_{asY}^{-1} & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -\tau_{asZ}^{-1} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & -\tau_{gYZa}^{-1} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & -\tau_{gZXa}^{-1} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & -\tau_{gXYa}^{-1} \end{bmatrix} \tag{9.77}$$

and the corresponding 21×21 covariance matrix

$$Q_S \stackrel{\text{def}}{=} \mathop{\mathrm{E}}_{t}\langle w_S(t) w_S^T(t)\rangle \tag{9.78}$$

$$= \begin{bmatrix} 2\sigma_{abX}^2/\tau_{abX} & 0 & 0 & \cdots & 0 \\ 0 & 2\sigma_{abY}^2/\tau_{abY} & 0 & \cdots & 0 \\ 0 & 0 & 2\sigma_{abZ}^2/\tau_{abZ} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 2\sigma_{gXYa}^2/\tau_{gXYa} \end{bmatrix}. \tag{9.79}$$

This is a model for independent drift of the sensor errors. However, a model based on empirical data may indicate correlations among the sensor errors that can be exploited to improve the performance of calibration and/or GNSS/INS integration.

The dynamic coefficient submatrix F_{SS} and the covariance matrix Q_S are computed by the MATLAB function `FSSQSmat.m` on the CD.

9.3.7 Dynamic Coupling of Sensor Errors and Navigation Errors

There is no first-order dynamic coupling of navigation errors into sensor errors, but sensor errors do couple directly into the time-derivatives navigation errors.

9.3.7.1 Navigation Error Dynamics Sensor errors are modeled in sensor-fixed coordinates. The INS resolves these errors from ISA-fixed *XYZ* coordinates to NED navigation coordinates and integrates them to increment the navigation errors. This mechanization results in a formula for the derivatives of navigation errors as a function of sensor errors.

The influence of the nine accelerometer errors and twelve gyroscope errors listed in Section 9.3.6.5 on six of the nine navigation errors listed in Section 9.3.5.2

can be modeled as

$$\frac{d}{dt}\begin{bmatrix}\dot{\varepsilon}_N\\ \dot{\varepsilon}_E\\ \dot{\varepsilon}_D\end{bmatrix} = C_{NED}^{XYZ}\left\{\begin{bmatrix}\varepsilon_{abX}\\ \varepsilon_{abY}\\ \varepsilon_{abZ}\end{bmatrix} + [\varepsilon_{aL}]a_{\text{sensed}}\right\} \tag{9.80}$$

$$\frac{d}{dt}\begin{bmatrix}\varepsilon_{\theta N}\\ \varepsilon_{\theta E}\\ \varepsilon_{\theta D}\end{bmatrix} = C_{NED}^{XYZ}\left\{\begin{bmatrix}\varepsilon_{gbX}\\ \varepsilon_{gbY}\\ \varepsilon_{gbZ}\end{bmatrix} + [\varepsilon_{gL}]\omega_{\text{sensed}}\right\} \tag{9.81}$$

$$[\varepsilon_{aL}] = \begin{bmatrix}\varepsilon_{asX} & \varepsilon_{aXYs} & \varepsilon_{aXZs}\\ \varepsilon_{aXYs} & \varepsilon_{asY} & \varepsilon_{aYZs}\\ \varepsilon_{aXZs} & \varepsilon_{aYZs} & \varepsilon_{asZ}\end{bmatrix} \tag{9.82}$$

$$[\varepsilon_{gL}] = \begin{bmatrix}\varepsilon_{gsX} & \varepsilon_{gXYs} - \varepsilon_{gXYa} & \varepsilon_{gXZs} + \varepsilon_{gXZa}\\ \varepsilon_{gXYs} + \varepsilon_{gXYa} & \varepsilon_{gsYs} & \varepsilon_{gYZs} - \varepsilon_{gYZa}\\ \varepsilon_{gXZs} - \varepsilon_{gXZa} & \varepsilon_{gYZs} + \varepsilon_{gYZa} & \varepsilon_{gsZs}\end{bmatrix}, \tag{9.83}$$

where C_{NED}^{XYZ} is the coordinate transformation matrix from ISA-fixed *XYZ* coordinates to NED coordinates, and the six affected INS navigation errors are

$\dot{\varepsilon}_N$, $\dot{\varepsilon}_E$ and $\dot{\varepsilon}_D$, the INS velocity errors.

$\varepsilon_{\theta N}$, $\varepsilon_{\theta E}$ and $\varepsilon_{\theta D}$, the INS tilt and heading errors.

The matrix C_{NED}^{XYZ} would be an identity matrix for gimbaled systems with sensor axes aligned to north, east, and down. For *alpha wander* gimbaled system implementations the ISA is rotated about the vertical axis through a known angle α, in which case

$$C_{NED}^{XYZ} = \begin{bmatrix}\cos(\alpha) & -\sin(\alpha) & 0\\ \sin(\alpha) & \cos(\alpha) & 0\\ 0 & 0 & 1\end{bmatrix}. \tag{9.84}$$

For a strapdown INS, C_{NED}^{XYZ} is part of the navigation solution.

The matrix ε_{aL} in Equation 9.82 is the error in the accelerometer scale factor and the misalignment error matrix due to sensor errors, and the vector a_{sensed} is the sensed acceleration vector in *XYZ* coordinates.

The matrix ε_{gL} in Equation 9.83 is the error in the gyroscope scale factor and the misalignment error matrix due to sensor errors, and ω_{sensed} is the sensed rotation rate vector in *XYZ* coordinates.

9.3.7.2 State Space Dynamic Model If the INS error model is to include the vector x_N of nine navigation errors and the vector x_S of 21 sensor errors (9 for accelerometers + 12 for gyroscopes), then the full error state vector will have

30 state variables:

$$x_{INS} = \begin{bmatrix} x_N & (9 \times 1) \\ x_S & (21 \times 1) \end{bmatrix} \tag{9.85}$$

$$x_N^T = [\varepsilon_N \quad \varepsilon_E \quad \varepsilon_D \quad \dot{\varepsilon}_N \quad \dot{\varepsilon}_E \quad \dot{\varepsilon}_D \quad \varepsilon_{\theta N} \quad \varepsilon_{\theta E} \quad \varepsilon_{\theta D}] \tag{9.86}$$

$$x_S = \begin{bmatrix} x_A & (9 \times 1) \\ x_G & (12 \times 1) \end{bmatrix} \tag{9.87}$$

$$x_A = [\varepsilon_{abX}, \quad \varepsilon_{abY}, \quad \varepsilon_{abZ}, \quad \varepsilon_{asX}, \quad \varepsilon_{asY}, \quad \varepsilon_{asZ}, \quad \varepsilon_{aYZs}, \quad \varepsilon_{aZXs}, \quad \varepsilon_{aXYs}] \tag{9.88}$$

$$x_G = [\varepsilon_{gbX}, \quad \varepsilon_{gbY}, \quad \varepsilon_{gbZ}, \quad \varepsilon_{gsX}, \quad \varepsilon_{gsY}, \quad \varepsilon_{gsZ}, \tag{9.89}$$

$$\varepsilon_{gYZs}, \quad \varepsilon_{gZXs}, \quad \varepsilon_{gXYs}, \quad \varepsilon_{gYZa}, \quad \varepsilon_{gZXa}, \quad \varepsilon_{gXYa}] \tag{9.90}$$

and the linear dynamic coupling between the two subvectors x_N and x_S expressed in Equations 9.80 and 9.81 can be rearranged into more conventional state-space form as

$$\frac{d}{dt}x_N = F_{NS}x_S \tag{9.91}$$

$$F_{NS} = \begin{bmatrix} 0_{3\times3} & 0_{3\times6} & 0_{3\times3} & 0_{3\times9} \\ C_{NED}^{XYZ} & F_{NS22} & 0_{3\times3} & 0_{3\times9} \\ 0_{3\times3} & 0_{3\times6} & C_{NED}^{XYZ} & F_{NS34} \end{bmatrix} \tag{9.92}$$

$$F_{NS22} = C_{NED}^{XYZ} \begin{bmatrix} a_X & 0 & 0 & 0 & a_Z & a_Y \\ 0 & a_Y & 0 & a_Z & 0 & a_X \\ 0 & 0 & a_Z & a_Y & a_X & 0 \end{bmatrix} \tag{9.93}$$

$$F_{NS34} = C_{NED}^{XYZ} \begin{bmatrix} \omega_X & 0 & 0 & 0 & \omega_Z & \omega_Y & 0 & \omega_Z & -\omega_Y \\ 0 & \omega_Y & 0 & \omega_Z & 0 & \omega_X & -\omega_Z & 0 & \omega_X \\ 0 & 0 & \omega_Z & \omega_Y & \omega_X & 0 & \omega_Y & -\omega_X & 0 \end{bmatrix}, \tag{9.94}$$

where a_X, a_Y, and a_Z are the acceleration components in XYZ coordinates, and ω_X, ω_Y, and ω_Z are the corresponding attitude rate components.

The dynamic coefficient submatrix F_{NS} is computed by the MATLAB function `FNSmat.m` on the CD.

Example 9.4 (Free Inertial Performance of Stationary INS) The MATLAB m-file `FreeInertial.m` on the CD uses the above models to calculate the propagation of INS navigation errors over a 10-h period, starting with zero navigation errors and the sensor error parameters shown in Table 9.2. These do not include sensor input axis misalignment errors.

TABLE 9.2 Sensor Error Parameters in Example 9.4

Inertial Sensor Type	Input Axis Direct.	Sensor Error Type	Sensor Error Symbol	RMS Variation σ	RMS Variation [units]	Correl. Time [min]
Accel.	Roll	Bias	ε_{abR}	40	μg	20
		S. Fact.	ε_{asR}	100	[ppm]	20
	Pitch	Bias	ε_{abP}	40	μg	20
		S. Fact.	ε_{asP}	100	[ppm]	20
	Yaw	Bias	ε_{abY}	40	μg	20
		S. Fact.	ε_{asY}	100	[ppm]	20
Gyro.	Roll	Bias	ε_{gbR}	.01	[deg/h]	20
		S. Fact.	ε_{gsR}	100	[ppm]	20
	Pitch	Bias	ε_{gbP}	.01	[deg/h]	20
		S. Fact.	ε_{gsP}	100	[ppm]	20
	Yaw	Bias	ε_{gbY}	.01	[deg/h]	20
		S. Fact.	ε_{gsY}	100	[ppm]	20

The simulation is for a stationary INS with its sensor axes aligned with NED coordinates, and with damping of the otherwise unstable vertical channel errors.

The resulting plot of RMS per axis horizontal error versus time is shown in Figure 9.13. The plot shows a sharp rise for the first $\approx$1 h of data but has a relatively constant slope thereafter. For that reason, it is not unusual to omit the first $\approx$1 h of data

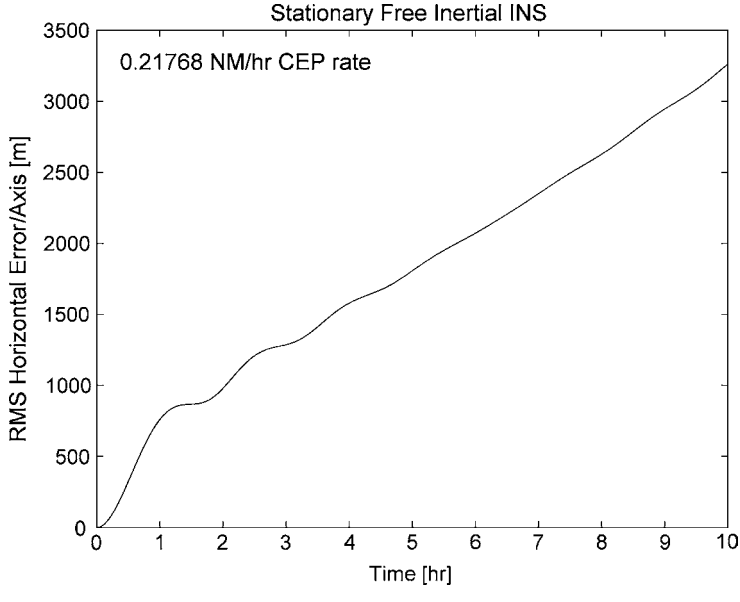


Figure 9.13 Stationary free inertial performance in Example 9.4.

in calculating the CEP rate. The least-squares CEP rate in this example is about 0.2 nautical miles per hour, but stationary navigation is not a good test of INS performance.

The conversion factor from RMS per axis horizontal error to CEP is about 1.18, and the nautical mile is defined as 1852 m in SI units.

9.3.8 Sensor-Aided INS

For INS applications lasting for hours or days, auxiliary sensors can be used to help keep INS navigation errors bounded. These auxiliary sensors may include any of the following:

1. Celestial navigation, such as
   (a) A sextant to obtain independent position measurements.
   (b) An INS-mounted star tracker to detect and correct tilts and heading errors.
2. A barometric altimeter for altitude measurements.
3. A water pressure sensor for measuring the depth of a submarine.
4. Radar (onboard or ground-based).
5. Radio navigation systems, such as
   (a) LORAN (long range aid to navigation).
   (b) VOR (VHF omnidirectional range).
   (c) DME (distance measuring equipment).
   (d) GNSS (global navigation satellite systems), including GPS (global positioning service).

An example implementation using GPS to stabilize altitude error in INS navigation is given in Section 9.6.1. Examples of GNSS/INS integration models are given in Sections 9.6 and 9.7.

9.4 GLOBAL NAVIGATION SATELLITE SYSTEMS (GNSS)

Pseudoranges GNSS uses satellites orbiting the earth as radio beacons for navigation, as illustrated in Figure 9.14. Signals transmitted by the satellites include time markers so that a receiver below can calculate the time of travel δt_j of the signals from the jth satellite to the receiver and use it to calculate the *pseudorange*

$$\rho_j \stackrel{\text{def}}{=} c \times \delta t_j \tag{9.95}$$

between that satellite and the receiver. The integer constant

$$c \stackrel{\text{def}}{=} 299,792,458\,\text{m/s}$$

is the speed of light in SI units.

Given the precise locations of the satellites, three or more pseudoranges from different satellites should suffice for determining the location of the receiver.

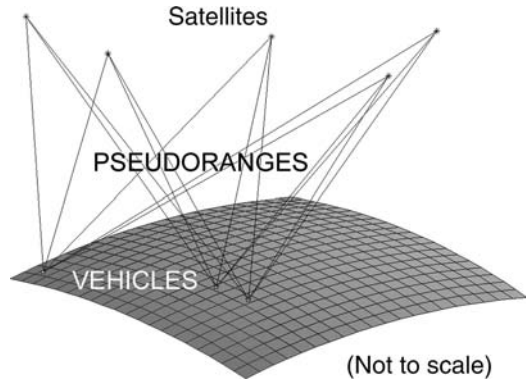


Figure 9.14 Satellite navigation geometry.

In practice, a less expensive receiver clock can be used if four or more pseudoranges are used for correcting receiver clock errors as well.

We have shown in Example 7.1 how GNSS receivers function as position sensors, with each satellite signal being tracked by the receiver providing a one-dimensional measure of receiver antenna position. The same example demonstrates how any approximation errors due to nonlinearities are insignificant compared to sensor noise.

Pseudorange Differences Backward differences $\rho_k - \rho_{k-1}$ of pseudoranges measured one time step ($\Delta t = 1$ second) apart are particularly useful in navigation for the following reasons:

1. They provide a relatively accurate measurement of the component of host vehicle velocity in the direction to the satellite.
2. These velocity measurements are not very sensitive to propagation delay errors, which have correlation times on the order of 1 min.
3. They are also not particularly sensitive to clock errors, the correlation times of which are usually much greater than the intersample interval.
4. Velocity measurements help to reduce filter lag. Without velocity measurements, it might take several filter cycles before estimates "catch up" to true values.

GNSS Design Analysis Soon after the Kalman filter was first introduced, it played a significant role in the development of GNSS technology. The Riccati equation from the Kalman filter was used in early design studies to predict how accurate the navigation solutions would be, and how this accuracy depends on design features such as satellite and receiver clock accuracies and satellite orbit geometry. The first fully functional GNSS implementation was the NAVSTAR GPS, developed by the U.S. Department of Defense from the early 1970s through the mid-1980s. It was made available for civilian use by order of President Ronald Reagan shortly after Soviet interceptor aircraft shot down Korean Air Lines Flight 007 when its inertial navigation system errors caused it to stray into Soviet airspace on September 1, 1983.

Kalman Filtering in GNSS The GPS software infrastructure has been called "one enormous Kalman filter." Kalman filters with hundreds of state variables are used to maintain accurate estimates of GNSS satellite trajectories, satellite clock corrections, and ionospheric propagation delay corrections.

GLONASS and Galileo GLONASS is an independent constellation of GNSS satellites with an essentially similar navigation function using different satellite signal characteristics. It was developed by the former Soviet Union and is maintained by Russia.[5] The European Commission is currently managing and funding efforts to develop a third, nonmilitary version of GNSS named Galileo.

9.4.1 System Operation

The primary function of a GNSS system is to provide accurate radio navigation signals to its receivers. This, in turn, requires maintaining the satellites in orbit, maintaining accurate knowledge of their locations at all times, monitoring and maintaining the accuracy of their onboard clocks, and maintaining up-to-date estimates of the signal propagation delays at all receiver locations. Approximate propagation delay information is broadcast as part of the satellite signals and used by the receivers to calculate the net delays at their locations. The signals also include satellite timing information used by the receivers to calculate the pseudoranges to all satellites being tracked.

For a more detailed description of the internal workings of the GPS infrastructure, see [106].

9.4.2 System Error Sources

GNSS system designers use an *error budget* of allowable subsystem error contributions for achieving a specified user navigation accuracy. This allows designers to achieve overall performance improvement where it is easiest (and/or least expensive) in order to compensate for errors elsewhere that may be more difficult to reduce. In similar fashion, receiver designers use their error budgets to allocate receiver-specific errors to achieve their own navigation accuracy goals. These error budgets may include hundreds of recognized contributions and their associated error models. The interface description for each GNSS system typically provides users with the resulting pseudorange performance statistics, which are sufficient for determining navigation performance for a given host vehicle and application.

9.4.2.1 Propagation Delay A dominant pseudorange error source is the smaller-scale variations in ionospheric propagation that are not adequately measured by the worldwide distribution of ground station monitors.

Two-Frequency Compensation The U.S. GPS system currently uses a second frequency channel with different propagation delays. Users with access to this

[5]The number of available satellites had declined but is currently (2007) being replenished.

channel can use the measured delay difference between the two channels to calculate the total propagation delay for each satellite signal. This second frequency channel is currently encrypted so that only U.S. military users can acquire it. However, the next generation of GPS satellites is designed to add a second channel for civilian use as well.

GLONASS and Galileo do not encrypt the secondary channel, but Galileo is currently being designed to deny access to all channels to all unauthorized (i.e., nonpaying) users.

Single-Frequency Compensation For receivers with no capability to acquire and track a second satellite frequency channel for calculating propagation delay errors, the RMS signal propagation delay uncertainties can be on the order of 10 m or more. They tend to be time-correlated, with correlation times on the order of 1 min.

A Kalman filter can be designed and used to estimate the uncompensated propagation delay errors. It uses an exponentially time-correlated model for each satellite signal, which requires one additional state variable for each satellite being used.

9.4.2.2 Clock Errors GNSS receivers need a frequency reference (clock) to acquire and track the signals from GNSS satellites. Each satellite signal is tracked using frequency- and phase-lock loops, and each satellite signal is then decoded to derive the timing and satellite location information used in providing the navigation solution. The navigation solution generally includes corrections for the phase and frequency of the clock, as well as the location of the receiver antenna. Most GNSS receivers use relatively inexpensive quartz clocks. After four or more satellites have been acquired, the additional information available for the navigation solution is used to estimate and compensate for errors in the clock.

A Kalman filter is used to estimate the host vehicle position, the receiver clock frequency (a process called *syntonization*), and relative clock phase error (called *synchronization*).

9.4.2.3 Other Signal Corruption Sources For host vehicles operating on the surface of the earth, the signals from satellites can be blocked by objects protruding above the horizon. These include natural objects such as mountainous terrain or trees and man-made objects such as tall buildings or overpasses.

The resulting signal corruption can be total blockage or interference from signals reflected off objects (including the surrounding ground or water surface). The presence of signals that have been deflected and delayed is called the *multipath problem*. For a relatively recent overview of signal processing countermeasures to the multipath problem, see [106].

9.4.3 GNSS Signal Tracking

GNSS receivers are designed to acquire and track the available signals from satellites (i.e., those that are currently in view). The number of satellite signals that can be tracked simultaneously varies from one receiver design to another, but most

current designs can track about a dozen satellites at once. For a more comprehensive coverage of receiver designs, see [106] and the references therein.

Most receivers are designed to cope with intermittent satellite signal loss by using a Kalman filter to carry the navigation solution forward in time without the missing signal(s), then using the resulting position solution for more rapid reacquisition when the signal is again available.

9.4.4 Modeled Error Sources for Receivers

Extended Kalman filters are used in most GNSS receivers to estimate receiver locations using measured pseudoranges. When the receiver is mounted on a mobile platform such as a land vehicle, ship, or aircraft, parameters of the Kalman filter can be optimized for tracking the vehicle motions.

Besides the unpredictable motions of the host vehicle, there are many other sources of error that can influence the accuracy of the estimates of position and velocity generated by the Kalman filter in the GNSS receiver. They include the following:

- *Errors in satellite position information broadcast by the satellites*. Satellite signals received at many ground-based receivers at known locations are used in Kalman filters (with very large state vectors) for estimating the satellite trajectories. This can keep the RMS satellite position errors on the order of centimeters.
- *Timing errors in the clocks aboard the satellites*. Much research and development effort has been directed to the development of highly stable space-based clocks for GNSS, and each satellite typically contains several such clocks to maintain high reliability. The compensation parameters for these clocks are estimated using Kalman filtering.
- *Signal propagation delays caused by the atmosphere*. In receivers capable of tracking two different carrier signals from each satellite, the delay difference between the two frequencies can be used to estimate the delay at each frequency. In receivers without dual frequency tracking, the delay can be partially compensated for at the receiver by using compensation formulas stored in the receiver, with time-varying coefficients broadcast in the satellite signals. The compensation coefficients are also generated by Kalman filters, using some of the same ground-station data used in tracking the satellites. For applications allowing an auxiliary local receiver at a known location to track the remaining propagation delay variations, RMS positions errors on the order of 1 m or less are achievable. Otherwise, the propagation delay variations are equivalent to several meters of RMS pseudorange error and have correlation times on the order of 1 min.
- *Noise in the received signals due to background electromagnetic radiation*. This can be compensated for to some degree by increasing the satellite signal power or by multiplying the signal by a spreading code to use a wider bandwidth. Studies of these effects are used in determining the satellite transmission power requirements for achieving GNSS navigation accuracy objectives.

- *Electronic noise in the satellites and receivers.* This has been countered, where necessary, by using low-noise electronic materials (e.g., gallium arsenide) for critical circuits.
- *Signal processing errors in the receiver.* These errors have been reduced significantly by developing implementations in higher-speed digital electronics with longer wordlengths.

Lever Arm Effects The location of the receiver antenna on the vehicle must be taken into account to avoid certain kinds of errors. The effective phase center of the receiver antenna is where the signals from all satellites are summed together, and this becomes the effective position solution generated by a GNSS receiver. It is also the point in the signal flow where the effective receiver clock error solution is valid. This distinction is important when the GNSS navigation solution is combined with the navigation solution from an onboard INS, which estimates the position of its own inertial sensors. The vector offset of the antenna location from the ISA is called the *antenna lever arm*.

The most accurate receiver implementations for GNSS navigation are able to use the relative phase information in the satellite carrier signals to obtain highly accurate pseudorange information. These implementations typically estimate propagation delays in individual signals as well. The most accurate of these approaches can yield relative RMS position errors on the order of centimeters or less. The more common low-cost receivers, which do not use relative carrier phase information, have RMS position errors on the order of 10 m. The more common low-cost approaches are the ones we present here.

9.5 KALMAN FILTERS FOR GNSS

GNSS receivers can use a dozen or so pseudorange measurements to estimated three components of position. An estimation problem with so much redundant yet noisy data is a natural application for the Kalman filter, which allows some of the information redundancy to be used in removing the effects of error sources in the overall system.

The resulting Kalman filters include the subsystem models described above. The dynamic coefficient matrices of the resulting Kalman filter have the general structure illustrated in Figure 9.15, with the three major subsystems being dynamically independent.

The size of the resulting model is the sum of the sizes of the subsystem models.

The number of state variables needed to model host vehicle dynamics depends on the application. An example with only two vehicle state variables is given in Section 9.5.5, along with another with nine state variables.

The clock model described in Section 9.5.1 has two state variables.

The number of state variables in the propagation delay model described in Section 9.5.2 varies with the number of satellites being tracked, limited by the number of tracking channels in the receiver. It may be as few as 4 or as many as 12.

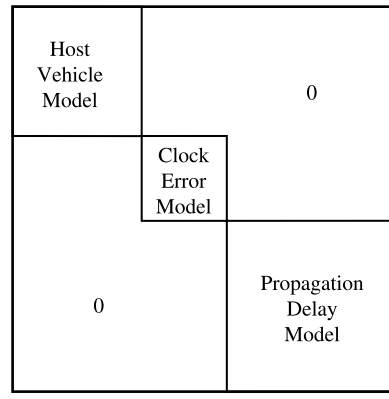


Figure 9.15 Structure of the Kalman filter model *F*-matrix.

9.5.1 Receiver Clock Correction

GNSS Clocks Some of the best modeling for real-world electromechanical systems has been done for clocks, and for quartz crystal oscillators (XO) in particular. The reference clocks for GNSS use a combination of high-quality temperature-controlled crystal oscillators (TCXO) and "atomic clocks" based on hyperfine quantum state transitions of atoms (e.g., in rubidium or cesium vapor). The TCXO provides short-term frequency stability to overcome poorer short-term coherence of the atomic clocks.

GNSS Receiver Clocks Most receiver clocks are microprocessor-controlled XO with little or no temperature control. These are relatively inexpensive and quite stable over periods of time on the order of 0–10 s. This works quite well for GNSS receiver applications, provided that receivers can use the timing information from hyperaccurate clocks on the GNSS satellites to maintain the required long-term stability and accuracy of their own clocks. The typical measurement update period for GNSS receivers is 1 s. This allows the receiver to track its own clock errors relatively accurately, which allows receiver designers to use less expensive clocks.

9.5.1.1 Clock Error Model

Clock Phase and Frequency Tracking The most common receiver clock frequency and phase tracking implementation uses what is called a *type 2 tracker* to keep the receiver clock synchronized to GNSS satellite clocks. This is essentially a Kalman filter with two state variables:

C_b, the receiver clock bias (i.e., offset from satellite time). The value in seconds can be scaled by the speed of light c to maintain C_b in distance units (e.g., meters).

C_d, the receiver clock drift rate, or time rate-of-change of bias. It can also be scaled by c to maintain C_d in velocity units.

Dynamic Model in Continuous Time The clock state vector is then two-dimensional, with its dynamic model in continuous time being

$$x = \begin{bmatrix} C_b \\ C_d \end{bmatrix} \tag{9.96}$$

$$\dot{x} = \underbrace{\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}}_{F} x + w(t) \tag{9.97}$$

$$w(t) \stackrel{\text{def}}{=} \begin{bmatrix} w_b(t) \\ w_d(t) \end{bmatrix} \tag{9.98}$$

$$Q_t \stackrel{\text{def}}{=} \mathop{\mathrm{E}}_{t}\left\langle w(t) w^T(t) \right\rangle \tag{9.99}$$

$$= \begin{bmatrix} q_{tbb} & 0 \\ 0 & q_{tdd} \end{bmatrix}. \tag{9.100}$$

That is, the zero-mean white noise processes $w_b(t)$ and $w_d(t)$ are uncorrelated.

This model is a short-term approximation of what is called *flicker noise* in clocks. The PSD of flicker noise as a function of frequency f falls off as $1/f$. This behavior cannot be modeled exactly by linear stochastic differential equations.

Dynamic Model in Discrete Time The equivalent model in discrete time will be

$$x_k = \underbrace{\begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}}_{\Phi(\Delta t)} x_{k-1} + w_{k-1} \tag{9.101}$$

$$\Phi(s) \stackrel{\text{def}}{=} \exp(Fs) \tag{9.102}$$

$$= \begin{bmatrix} 1 & s \\ 0 & 1 \end{bmatrix} \tag{9.103}$$

$$w_{k-1} \stackrel{\text{def}}{=} \begin{bmatrix} w_{b,\,k-1} \\ w_{d,\,k-1} \end{bmatrix} \tag{9.104}$$

$$Q_{k-1} \stackrel{\text{def}}{=} \mathop{\mathrm{E}}_{k}\left\langle w_{k-1} w_{k-1}^T \right\rangle \tag{9.105}$$

$$= \Phi(\Delta t)\left[\int_0^{\Delta t} \Phi^{-1}(s) Q_t(s) \Phi^{-T}(s)\, ds\right] \Phi^{\mathrm{T}}(\Delta t) \tag{9.106}$$

$$= \begin{bmatrix} q_{tbb}\Delta t + q_{tdd}\Delta t^3/3 & q_{tdd}\Delta t^2/2 \\ q_{tdd}\Delta t^2/2 & q_{tdd}\Delta t \end{bmatrix}, \tag{9.107}$$

where Equation 9.106 is from Equation 3.130, which we use to calculate Q_{k-1} from Q_t.

Covariance Propagation in Discrete Time The covariance propagation between updates in discrete time has the form

$$P_k = \Phi(\Delta t) P_{k-1} \Phi^T(\Delta t) + Q_{k-1} \tag{9.108}$$

$$Q_{k-1} = \begin{bmatrix} q_{bb} & q_{bd} \\ q_{db} & q_{dd} \end{bmatrix} \tag{9.109}$$

$$q_{bb} = q_{tbb}\Delta t + q_{tdd}\Delta t^3/3 \tag{9.110}$$

$$q_{bd} = q_{tdd}\Delta t^2/2 \tag{9.111}$$

$$q_{db} = q_{bd} \tag{9.112}$$

$$q_{dd} = q_{tdd}\Delta t. \tag{9.113}$$

Representative Process Noise Covariance Values The values of mean-squared bias noise q_{bb} and drift noise q_{dd} vary with the quality (and price) of the clock used. A common statistic used for the quality of a clock is its RMS relative frequency stability over a stated time period. We show here a method for calculating Q_t and Q_{k-1} from the quoted RMS relative frequency stability for a time period equal to the intersample period (1 s). The same method can be used for any RMS relative frequency stability and intersample period.

Example 9.5 (Calculating Clock Model Disturbance Noise) Reasonably low-cost quartz crystal clocks have frequency stabilities on the order of $10^{-9}-10^{-6}$ parts per part over the time between GPS pseudorange measurements (1 s). That is, for $f =$ clock frequency, the RMS incremental change in relative frequency $[f(t) - f(t-1)]/f(t-1)$

$$\frac{\sigma_f}{f} \approx 10^{-9} \text{ to } 10^{-6}. \tag{9.114}$$

We will use the lower value (10^{-9}) to demonstrate a methodology for translating this number into an equivalent process noise covariance. However, the result scales with the square of the clock stability value, so it can be scaled to any stability figure. For example, the value for $\sigma_f/f = 10^{-7}$ will be

$$Q_{10^{-7}} = \left(\frac{10^{-7}}{10^{-9}}\right)^2 Q_{10^{-9}}, \tag{9.115}$$

where $Q_{10^{-9}}$ is the equivalent process noise covariance for $\sigma_f/f = 10^{-9}$.

If we convert the clock model parameters to velocity units, then

$$q_{dd} = (c\,\sigma_f/f)^2 \tag{9.116}$$

$$\approx (3 \times 10^8 \times 10^{-9})^2 \tag{9.117}$$

$$\approx 0.1\ [\mathrm{m}^2/\mathrm{s}^2] \tag{9.118}$$

for an intersample period of $\Delta t = 1$ s.

From these values of q_{dd} and Δt, one can solve for the equivalent process noise covariance q_{tdd} in continuous time as

$$q_{tdd} = q_{dd}/\Delta t \tag{9.119}$$

$$= 0.1\ [\mathrm{m}^2/\mathrm{s}^3]. \tag{9.120}$$

The value of frequency drift variance q_{tdd} depends primarily on the quality of the quartz crystal, its temperature control, and the stability of its associated control electronics. The value of the phase noise variance q_{tbb} depends more on the electronics. In a "balanced" design, the contributions of both to mean-squared timing errors are about equal. If we assume that the contributions to q_{bb} from q_{tbb} and q_{tdd} are about equal, then

$$q_{tbb}\Delta t \approx q_{tdd}\Delta t^3/3 \tag{9.121}$$

$$q_{tbb} \approx q_{tdd}\Delta t^2/3 \tag{9.122}$$

$$\approx 0.03\ [\mathrm{m}^2/\mathrm{s}] \tag{9.123}$$

$$Q_t \approx \begin{bmatrix} 0.03 & 0 \\ 0 & 0.10 \end{bmatrix}, \tag{9.124}$$

and

$$q_{bb} = q_{tbb}\Delta t + q_{tdd}\Delta t^3/3 \tag{9.125}$$

$$\approx 0.06\ [\mathrm{m}^2] \tag{9.126}$$

$$q_{bd} = q_{tdd}\Delta t^2/2 \tag{9.127}$$

$$\approx 0.05\ [\mathrm{m}^2/\mathrm{s}] \tag{9.128}$$

$$Q_{k-1} \approx \begin{bmatrix} 0.06 & 0.05 \\ 0.05 & 0.10 \end{bmatrix} \tag{9.129}$$

in distance and velocity units.

A plot of clock estimation uncertainties versus clock stability is shown in Figure 9.16 for a stationary receiver with good satellite geometry. Under such ideal conditions, clock stability does not severely compromise location uncertainty, but it does compromise clock syntonization (frequency tracking). This tends to corrupt the navigation solution, as well, when the receiver antenna is moving.

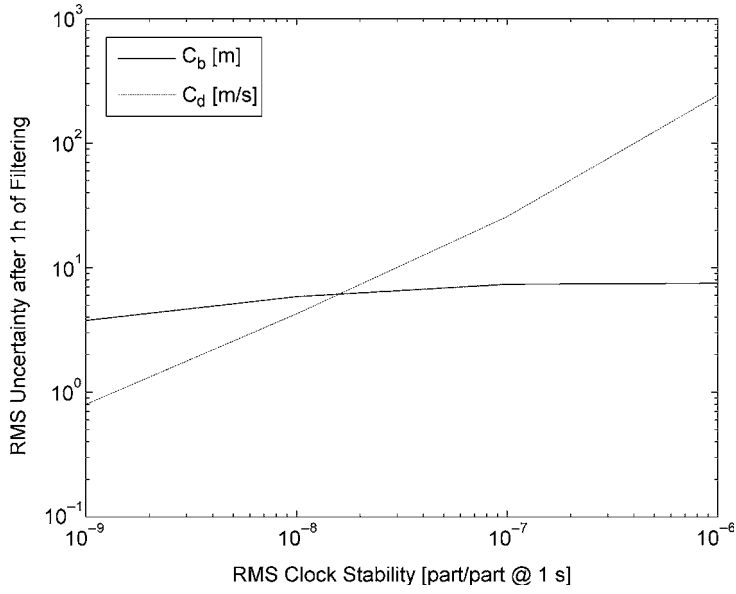


Figure 9.16 Clock synchronization and syntonization uncertainties.

Measurement Sensitivity Matrix The sensitivity of any pseudorange ρ to the clock state vector has the form

$$H_{\text{clock}} = \frac{\partial \rho}{\partial\, C_b,\ C_d} \tag{9.130}$$

$$= [1 \quad 0] \tag{9.131}$$

when C_b has units of distance. That is, a clock bias error ε_b is equivalent to a uniform increase of ε_b m in *all* pseudoranges simultaneously.

9.5.2 Signal Propagation Delay Models

The speed of transmission of GNSS signals through the atmosphere is not constant. Due to temporal and spatial variations in density, water vapor, and ionization, it varies with time and location. Relative variations in the speed of propagation are small, but the effects can produce errors of hundreds of meters in the navigation solution. For GPS, each satellite transmits signals at two different carrier frequencies so that the differential delay between the two carriers can be used to calculate the average delay as well. However, the second GPS carrier signal is currently encrypted so that only U.S. military receivers can use it.

For GPS receivers without two-frequency capability, a coarse map of local delay corrections is transmitted as part of the signals from the satellites. This map is created by using a network of ground stations for monitoring propagation delays. It reduces the propagation delays to the order of 10 m, with correlation times on the order of 1 min.

To further reduce the errors due to propagation delays, each receiver can use a Kalman filter to estimate and compensate for the residual time-correlated

pseudorange errors in each satellite signal. The delays are approximated as exponentially correlated processes, the model for which in continuous time is

$$\dot{\Delta}\rho = -\Delta\rho/\tau + w(t) \tag{9.132}$$

$$w(t) \in \mathcal{N}(0, Q_t) \tag{9.133}$$

$$Q_t = 2\sigma^2/\tau \tag{9.134}$$

$$\sigma \approx 10 \text{ m}$$

$$\tau \approx 60 \text{ s}.$$

The equivalent pseudorange error model in discrete time is

$$\Delta\rho_k = \Phi\Delta\rho_{k-1} + w_{k-1} \tag{9.135}$$

$$\Phi = \exp(-\Delta t/\tau) \tag{9.136}$$

$$\Delta t = \text{discrete time step}$$

$$\tau \approx 60 \text{ s}$$

$$w_k \in \mathcal{N}(0, Q) \tag{9.137}$$

$$Q = \sigma^2(1 - \Phi^2) \tag{9.138}$$

$$\sigma \approx 10 \text{ m}.$$

9.5.2.1 Measurement Vector The navigation state vector must be augmented by adding one such state variable for each satellite being used in the navigation solution. This can be implemented by using one such state variable for each GNSS satellite and setting the corresponding row of the measurement sensitivity matrix to zero whenever that satellite is not being used for navigation.

With a little more programming effort, the state variable can also be implemented so that the state vector always has the minimum required dimension. As satellites go out of view, their associated pseudorange error variables can be removed from the state vector. When new satellites are added, their pseudorange error variables can be inserted in the state vector, with associated variances initialized at σ^2. So long as individual satellites remain unused for at least a few minutes, this implementation does not compromise performance.

9.5.3 Serial Processing of Pseudorange Measurements

GNSS receivers track the independent signals from different satellites using a dedicated signal tracking loop for each satellite, but the calculated pseudoranges are output at the same time epoch t_k. The resulting pseudorange measurement vector,

$$z_k = \begin{bmatrix} \rho_k^{[1]} \\ \rho_k^{[2]} \\ \rho_k^{[3]} \\ \vdots \\ \rho_k^{[\ell]} \end{bmatrix} + \begin{bmatrix} v_k^{[1]} \\ v_k^{[2]} \\ v_k^{[3]} \\ \vdots \\ v_k^{[\ell]} \end{bmatrix}, \tag{9.139}$$

has additive noise that is correlated over time $\{t_k\}$, but uncorrelated between satellites.

9.5.3.1 Measurement Noise Covariance After the time-correlated errors have been removed by filtering, the remaining errors are essentially independent zero-mean white noise processes with a scalar covariance matrix,

$$R_k \stackrel{\text{def}}{=} \mathrm{E}\left\langle \begin{bmatrix} v_k^{[1]} \\ v_k^{[2]} \\ v_k^{[3]} \\ \vdots \\ v_k^{[\ell]} \end{bmatrix} \begin{bmatrix} v_k^{[1]} & v_k^{[2]} & v_k^{[3]} & \cdots & v_k^{[\ell]} \end{bmatrix} \right\rangle \tag{9.140}$$

$$= \begin{bmatrix} \sigma^2 & 0 & 0 & \cdots & 0 \\ 0 & \sigma^2 & 0 & \cdots & 0 \\ 0 & 0 & \sigma^2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \sigma^2 \end{bmatrix}. \tag{9.141}$$

9.5.3.2 Avoiding Matrix Inversion Because R_k of Equation 9.141 is a diagonal matrix, the individual pseudorange measurements

$$z_k^{[j]} = \rho_k^{[j]} + v_k^{[j]} \tag{9.142}$$

can be processed by the Kalman filter as ℓ independent scalar measurements with variances σ_j^2 and measurement sensitivity matrices equal to the rows of H_k:

$$H_k = \begin{bmatrix} h_k^{[1]} \\ h_k^{[2]} \\ h_k^{[3]} \\ \vdots \\ h_k^{[\ell]} \end{bmatrix}. \tag{9.143}$$

The matrix Riccati equation for the vector measurement update,

$$\overline{K}_k = P_k H_k^T \underbrace{\left[H_k P_k H_k^T + R_k\right]^{-1}}_{\text{inverse}} \tag{9.144}$$

$$P_k = P_k - \overline{K}_k H_k P_k, \tag{9.145}$$

requires a matrix inverse. This can be replaced by ℓ scalar updates

```
for j = 1:ℓ
   h   = H(j,:);
   PhT = P*h';
   K   = PhT/(h*PhT+ sigmasquared);
   P   = P - K*PhT';
end;
```

which avoids taking any matrix inverses in the Kalman filter implementation, decreases the amount of computation required, and reduces the influence of roundoff errors.

9.5.4 MATLAB Simulation Models

We use simulators of the host vehicle dynamics and GNSS satellite motions to demonstrate the different Kalman filter implementations for GNSS. These are described in the following subsections.

9.5.4.1 Host Vehicle Simulators

Stationary Receiver Stationary GNSS receivers are often used as clocks. Unless the precise receiver antenna location is already known, it can be estimated along with the navigation solution. The Kalman filter for this is relatively simple, and is used to demonstrate the effects of different error sources without the corrupting effects of host vehicle dynamics.

Racetrack Simulator A common host vehicle model used for simulations is for a vehicle running at 90 kph average speed on a figure-8 track of length 1.5 km, making one complete circuit every minute. The track's orientation and shape are illustrated in Figure 9.17. The limited position excursions of the track can be exploited by solving the navigation solution in locally level coordinates.

A vehicle running at 90 kph hardly qualifies as a race car, but integrated GNSS/INS navigation is used fairly commonly on all race cars in televised races on closed-circuit tracks—not for letting the drivers know where they are, but for letting the broadcast television system know where each race car is at all times. The onboard navigation solutions are telemetered to the track television recording system and used to generate on-screen graphics to point out individual cars during the race. The accuracy of the navigation solutions is generally on the order of 1 m or less, which is sufficient for this purpose. The heading and pitch angles of the optical

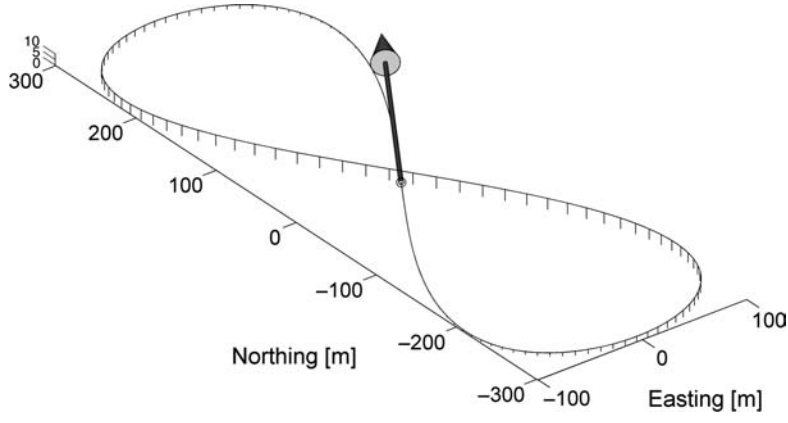


Figure 9.17 Figure-8 trajectory of length 1500 m.

TABLE 9.3 Vehicle Dynamics on Figure-8 Track

Statistic	Value	Unit	Statistic	Value	Unit
RMS N-S position excursion	212.9304	m	RMS delta velocity down*	0.0391	m/s
RMS E-W position excursion	70.9768	m	RMS delta roll rate*	0.0028	rad/s
RMS Vert. position excursion	3.5361	m	RMS delta pitch rate*	0.0077	rad/s
RMS N-S velocity	22.3017	m/s	RMS delta yaw rate*	0.0245	rad/s
RMS E-W velocity	14.8678	m/s	North position correlation time	13.4097	s
RMS vertical velocity	0.37024	m/s	East position correlation time	7.6696	s
RMS N-S acceleration	2.335	m/s/s	Vertical position corr. time	9.6786	s
RMS E-W acceleration	3.1134	m/s/s	North velocity correlation time	9.6786	s
RMS vertical acceleration	0.038778	m/s/s	East velocity correlation time	21.4921	s
RMS roll rate	0.0257	rad/s	Vertical velocity correlation time	13.4097	s
RMS pitch rate	0.0471	rad/s	North acceleration correlation time	13.4097	s
RMS yaw rate	0.1579	rad/s	East acceleration correlation time	7.6696	s
RMS delta velocity north*	2.3537	m/s	Vertical acceleration correlation time	9.6786	s
RMS delta velocity east*	3.1339	m/s			

*Sampled at 1-s intervals.

axes of track television cameras are used together with camera lens focal lengths to compute where each race car would appear on the recorded image, and this information is then used to generate text and pointers to locate and identify selected cars on the image during the race. The host vehicle dynamics in this application can be simplified to a two-dimensional model with along-track and cross-track components. Vehicle altitude will always be a known function of these two location components. We will not use this level of sophistication in our demonstrations, even though the figure-8 track location model uses only along-track position.[6]

Racetrack Dynamics The averaged vehicle dynamic conditions during figure-8 track simulations have the values shown in Table 9.3. To provide empirical values of for Q in the Kalman filter models for GNSS navigation, the RMS velocity

[6]Table 10.4 of [106] shows how this level of modeling can achieve RMS position uncertainties that are 30–50 times smaller than those from using more conventional host vehicle models.

changes were sampled at the GNSS navigation update intervals of 1 s on track simulations generated by the MATLAB function `Fig8TrackSim` on the CD. The other statistics were evaluated using 100 samples per second.

These values can be used for the parameters of the host vehicle dynamic models in Table 9.1.

9.5.4.2 GNSS Simulation Models

Fixed Satellite Locations These models are used in the MATLAB scripts in `Example0901.m` (to remove the effects of satellite motion) and `Example0906.m` (to demonstrate the influence of satellite geometry on performance) on the CD.

Real-World Satellite Motions This GNSS simulator uses the GPS satellite configuration from Wednesday, March 8, 2006, at 10:48 A.M., as downloaded from the U.S. Coast Guard site www.navcen.uscg.gov-/ftp/GPS/almanacs/yuma. There were 29 operational GPS satellites at that time.[7] The MATLAB script `YUMAdata.m` contains the GPS ephemeris information at that time, and it produces two global arrays used for GPS simulations. The MATLAB function `HSatSim.m` uses these global arrays to generate pseudorange measurement sensitivity matrices for GNSS receivers at a given latitude, longitude, and time.

9.5.5 Navigation Solutions

In GNSS navigation, the *navigation solution* is an estimate of the location of the antenna of a GNSS receiver.

The "front end" of the GNSS receiver generates the pseudorange measurements to all GNSS satellites being tracked. The digital processors in the "back end" of the GNSS receiver use Kalman filters to estimate the navigation solution, given the satellite locations (broadcast in each satellite signal) and the pseudoranges obtained by tracking the satellite signals. The navigation solution always includes the location of the receiver antenna with respect to the earth, but may include the solution for other "nuisance variables" as well. The full navigation solution may include any of the following:

1. The longitude, latitude, and altitude of the receiver antenna with respect to a specified datum (reference geoid for the shape of the earth).
2. Universal Time Coordinated (UTC), the international standard time. UTC is referenced to a worldwide collection of atomic clocks, but is adjusted occasionally by adding or deleting "leap seconds" to remain phase-locked to the rotation of the earth within $\approx \pm 7.5$ arc-seconds. (GNSS clock time is never subjected to such discrete changes for fear of what would happen to navigation solutions with pseudorange jumps of 299792458 m.)

[7]Currently (2007), an effort is underway within the civilian user community to make the standard GPS constellation include 30 satellites.

3. The receiver clock bias (difference from GNSS clock time), which is generally estimated as part of the navigation solution.
4. The receiver clock frequency, also generally estimated.
5. The velocity of the host vehicle, which is commonly estimated unless the receiver antenna is known to be stationary.
6. The acceleration of the host vehicle, which could be of interest in some applications (e.g., atmospheric braking of test vehicles).
7. Time-correlated pseudorange errors, due principally to smaller-scale variations in ionospheric delay. The RMS magnitude of these errors is on the order of 10 m, and the correlation times are on the order of 1 min, typically. Estimating these nuisance variables requires one additional Kalman filter state variable for each satellite being used in the navigation solution—which can be on the order of a dozen additional state variables.

The resulting receiver Kalman filter state vector dimension can range from 5 to more than 20 if receiver clock biases and frequencies are always included in the navigation solution. With additional state variables for the propagation delay corrections, the state vector dimension may exceed 30.

9.5.5.1 Effects of Satellite Geometry The following example uses a simple Kalman filter model with five state variables: three for position and two for clock error corrections. The associated Riccati equation solution is used to show that relative satellite locations influence position uncertainty in the Kalman filter estimate.

Example 9.6 (Effects of Satellite Geometry) This simplified example uses a fixed antenna location and fixed satellite geometry to demonstrate how navigation performance depends on satellite geometry—a phenomenon called *dilution of precision* (DOP).[8]

Measured pseudoranges from four satellites are used—the minimum number of satellites required for estimating position and clock errors. The dependence of observability on satellite geometry is demonstrated by using different sets of directions to these four satellites.

The model uses the two-state clock error model from Example 9.5 and three antenna position coordinates in locally level coordinates, for a total of five state variables:

$$x^T = [N \quad E \quad D \quad C_b \quad C_d], \tag{9.146}$$

where

N = north position in meters
E = east position in meters

[8]DOP is further subdivided into how it affects estimates of position, timing, etc. For more about DOP, see [101] or search for articles on the subject on the World Wide Web.

D = downward position in meters
C_b = receiver clock bias in meters
C_d = receiver relative clock drift rate in meters per second

The locally level coordinates [N, E, D] of antenna position are unknown constants, so that the upper-left 3×3 submatrix of Φ is an identity matrix and the upper-left 3×3 submatrix of Q is zero. The lower-right 2×2 submatrix of Q is consistent with a receiver clock with 10^{-8} parts per part RMS frequency stability over $\Delta t = 1$ s. That is,

$$Q = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 6 & 5 \\ 0 & 0 & 0 & 5 & 10 \end{bmatrix} \tag{9.147}$$

$$\Phi = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{9.148}$$

We further assume that

- RMS pseudorange measurement error is 15 m,
- Initial RMS antenna position uncertainty is 1 km,
- Initial RMS clock bias is 3 km (10 μs), and
- Initial RMS relative frequency uncertainty is 30 m/s (10^{-7} part/part),

so that the covariance matrices

$$R = \begin{bmatrix} 225 & 0 & 0 & 0 \\ 0 & 225 & 0 & 0 \\ 0 & 0 & 225 & 0 \\ 0 & 0 & 0 & 225 \end{bmatrix} \tag{9.149}$$

$$P_0 = \begin{bmatrix} 10^6 & 0 & 0 & 0 & 0 \\ 0 & 10^6 & 0 & 0 & 0 \\ 0 & 0 & 10^6 & 0 & 0 \\ 0 & 0 & 0 & 9 \times 10^6 & 0 \\ 0 & 0 & 0 & 0 & 900 \end{bmatrix}. \tag{9.150}$$

The off-diagonal values of R in Equation 9.149 are zero because there is effectively no correlation between pseudorange errors to different GNSS satellites. Except for receiver clock errors, the pseudorange measurement error mechanisms are effectively statistically independent between one satellite and another. Because the clock errors are part of the state vector, the remaining errors are uncorrelated.

The Kalman filter model in discrete time is completely defined by Φ_k, Q_k, H_k, and R_k, and performance depends on P_0 as well. In this example, H will depend on the directions to the four satellites. If we let the direction to the jth satellite be specified by its azimuth θ_j (measured clockwise from north) and elevation angle ϕ_j (measured upward from the horizon), then

$$H = \begin{bmatrix} -\cos(\theta_1)\cos(\phi_1) & -\sin(\theta_1)\cos(\phi_1) & \sin(\phi_1) & 1 & 0 \\ -\cos(\theta_2)\cos(\phi_2) & -\sin(\theta_2)\cos(\phi_2) & \sin(\phi_2) & 1 & 0 \\ -\cos(\theta_3)\cos(\phi_3) & -\sin(\theta_3)\cos(\phi_3) & \sin(\phi_3) & 1 & 0 \\ -\cos(\theta_4)\cos(\phi_4) & -\sin(\theta_4)\cos(\phi_4) & \sin(\phi_4) & 1 & 0 \end{bmatrix} \tag{9.151}$$

These equations are programmed in the MATLAB program `Example0906.m`, which allows you to enter four directions to GNSS satellites, then computes and plots navigation performance in terms of RMS uncertainties in position (three components) and clock errors (two parameters) versus time for 1 min.

Figures 9.18 and 9.19 were generated using this program. These are plots of the RMS uncertainty in the clock bias (C_b) and clock drift (C_d) versus time. The first of these uses good satellite geometry and the second uses bad satellite geometry.

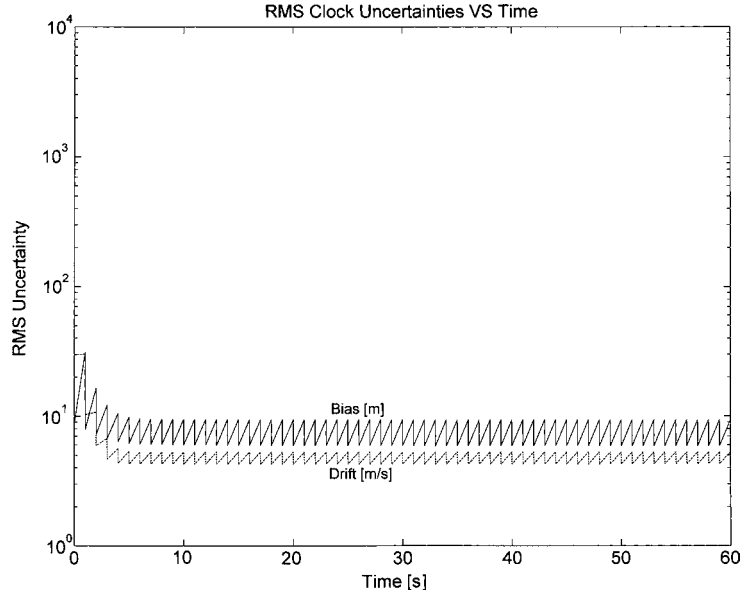


Figure 9.18 Clock parameter uncertainties with good satellite geometry.

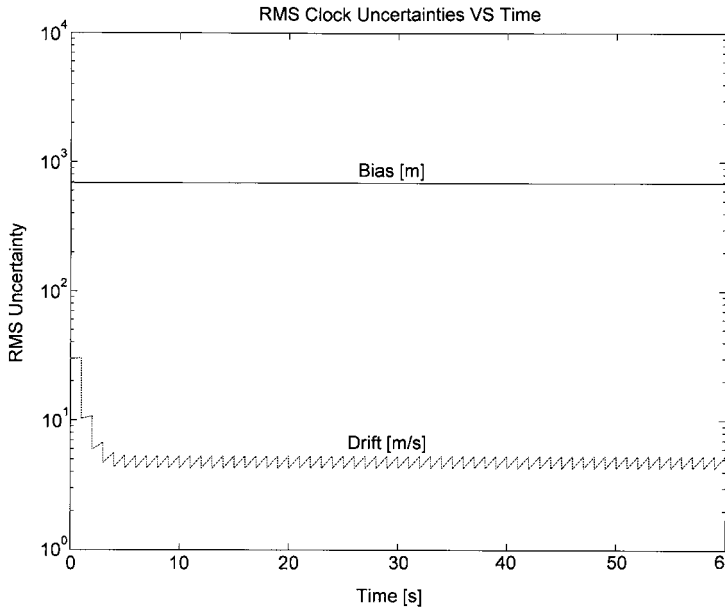


Figure 9.19 Clock parameter uncertainties with bad satellite geometry.

The good satellite geometry has three satellites equally space 120° apart in the horizontal plane, which is sufficient by itself to determine horizontal position *and* receiver clock time error. A receiver clock time error is equivalent to lengthening or shortening all pseudoranges the same amount, and the three horizontal pseudoranges are sufficient for determining horizontal position and clock time error simultaneously with this geometry. The fourth satellite is overhead, the pseudorange of which is sensitive only to altitude and clock time error. However, clock time error is already uniquely determinable from the first three pseudoranges.

The bad satellite geometry has all four satellites at 45° elevation, equally spaced 90° apart in azimuth. In that configuration, a clock timing error is not distinguishable from an antenna altitude error.

9.5.5.2 Effects of Host Vehicle Models The script `GNSSshootoutNCE.m`[9] on the CD compares side-by-side three different host vehicle dynamic models for the same navigation problem, which is the figure-8 track model described in Section 9.5.4.1 with time-correlated random vehicle velocity variations. The three host vehicle dynamic models are

MODL3, which uses model number 3 (type 2 tracker) from Table 9.1 with parameter values from Table 9.3. This vehicle model has six state variables.

[9]NCE stands for "no clock errors."

TABLE 9.4 GNSS Navigation with Different Vehicle Models and No Clock Errors

Vehicle Dynamic Model	Number of State Variables	RMS Position Error [m]			RMS Velocity Error [m/s]		
		North	East	Vert.	North	East	Vert.
MODL3	6	42.2	41.3	6.0	16.3	16.8	0.4
MODL5	6	22.9	25.1	3.5	19.8	15.4	0.4
MODL6	9	7.4	11.6	3.3	22.1	17.5	0.4

MODL5, which uses model number 5 (bounded RMS velocity) from Table 9.1 with parameter values from Table 9.3. This vehicle model has six state variables.

MODL6, which uses model number 6 (bounded RMS position) from Table 9.1 with parameter values from Table 9.3. This vehicle model has nine state variables.

Each model was "tuned" to the simulated host vehicle dynamics by setting its independent model parameters to match the values listed in Table 9.3. In all cases, the full Kalman filter model includes state variables for the time-correlated propagation delay errors *but does not include clock errors* (two state variables).

Clock errors were excluded to better demonstrate the influence of host vehicle dynamic modeling on estimating the dynamic state variables. The respective achievable performance values after taking into account the receiver clock errors will depend on the quality of the clock but will generally be worse in all cases. All results do include simulated time-correlated propagation delay errors and realistic GPS satellite trajectories.

The resulting respective RMS position and velocity errors from `GNSSshootoutNCE.m` are listed in Table 9.4. The values given in the table for the number of state variables are for the vehicle dynamic model only. The number of additional state variables required for propagation delay errors was the number of satellites in view ($\geq 15^\circ$ above the horizon), which varied from 9 to 11 during 2 h of simulated vehicle and satellite trajectories. Clock error modeling would add two more state variables to the model.

These results clearly indicate that host vehicle dynamic modeling can be important, although the differences would have been less dramatic if clock errors had been included.

The program `GNSSshootoutNCE.m` includes the MATLAB models for all cases, plus a specialized dynamic model (**FIG8**) designed specifically for the figure-8 track. The program also generates plots of position and velocity errors and their frequency spectra.

9.5.6 Using Pseudorange Differences

By exploiting known error characteristics in the signals, GNSS users have devised these methods (and many more) to improve GNSS navigation performance.

9.5.6.1 Time Differences GNSS navigation solutions can use the difference between successive pseudorange measurements to the same satellite as a measure of the velocity difference between the satellite and the receiver antenna. Because satellite velocities are known very accurately, and because the differences are relatively insensitive to signal propagation delays, these measurements are useful for estimating host vehicle velocities.

9.5.6.2 Spatial Differences Pseudoranges from separate antennas are also used in GNSS navigation:

1. Receiver antennas at known, fixed locations are used to estimate pseudorange errors to each available satellite due to uncompensated propagation delays. The propagation delays do not vary significantly over distances of tens or hundreds of kilometers. The estimated pseudorange corrections are broadcast to nearby receivers and used to improve the pseudorange measurement accuracies for each satellite. This approach can provide position accuracies on the order of centimeters.
2. Arrays of antennas rigidly mounted on a common base are used to estimate attitude. The GNSS receivers in this application commonly use carrier phase interferometry to provide a better measurement of attitude than pseudoranges alone. Antenna separations on the order of 1 m can provide RMS attitude accuracies of 1 millirad or less.

9.5.7 Derived Attitude Estimates

Although GNSS pseudorange measurements at a single antenna are insensitive to vehicle attitude, the estimated velocity and acceleration are commonly used to generate secondary estimates of heading, pitch angle, and roll angle. Here, these secondary estimates assume that the vehicle roll axis is aligned with the direction of velocity and that turns are coordinated. That is, the sensed acceleration is parallel to the vehicle yaw axis.

More sophisticated models might include the modeled effects of slide-slip (the angle between the velocity vector and the vehicle roll axis due to accelerations along the vehicle pitch axis), aerodynamic disturbances, and suspension reactions. Here, these other effects are ignored, except perhaps as part of the RMS uncertainties in the secondary estimates.

The simplified attitude estimation model is

$$\tan(\hat{\theta}_Y) = \frac{\hat{V}_E}{\hat{V}_N} \text{ (heading angle)} \tag{9.152}$$

$$\sin(\hat{\theta}_P) = \frac{-\hat{V}_D}{\sqrt{\hat{V}_N^2 + \hat{V}_E^2 + \hat{V}_D^2}} \text{ (pitch angle)} \tag{9.153}$$

$$\sin(\hat{\theta}_R) = \frac{U_D}{\cos(\hat{\theta}_P)} \text{ (roll angle)} \tag{9.154}$$

$$U = \frac{\hat{V} \otimes \left[I - \frac{\hat{V}\hat{V}^T}{\hat{V}^T\hat{V}}\right]\hat{A}}{\left|\hat{V} \otimes \left[I - \frac{\hat{V}\hat{V}^T}{\hat{V}^T\hat{V}}\right]\hat{A}\right|}, \tag{9.155}$$

where $\hat{V}$ is the estimated velocity vector and $\hat{A}$ is the estimated sensed acceleration vector (i.e., not including gravitational acceleration) in NED coordinates, and $\hat{\theta}_Y$, $\hat{\theta}_P$ and $\hat{\theta}_R$ are the estimated vehicle heading, pitch, and roll angles.

This model is good only so long as the magnitude of velocity is much greater than the RMS velocity uncertainty.

Results from a GNSS-only simulation on a figure-8 track are plotted in Figure 9.20. The simulated velocity magnitude $|V| \approx 25\,\mathrm{m/s}$ and its standard deviation $\sigma V \approx 7.7\,\mathrm{m/s}$ per axis for those periods of time when GNSS signals are available. The equivalent RMS derived angle uncertainties due to velocity uncertainties only would be on the order of $\sigma_\theta \approx 0.3\,\mathrm{rad} \approx 17^\circ$. This is much worse than the short-term RMS tilt uncertainties in INS navigation and is not very useful for dead reckoning either.

However, the angular uncertainties scale roughly as the RMS velocity uncertainty and the inverse of speed. Given the same GNSS velocity estimation uncertainties, vehicles on a freeway with $|V| \approx 100\,\mathrm{m/s}$ would have attitude uncertainties on the order of $4^\circ - 5^\circ$. For high-performance aircraft with $|V| \approx 300\,\mathrm{m/s}$, the RMS attitude

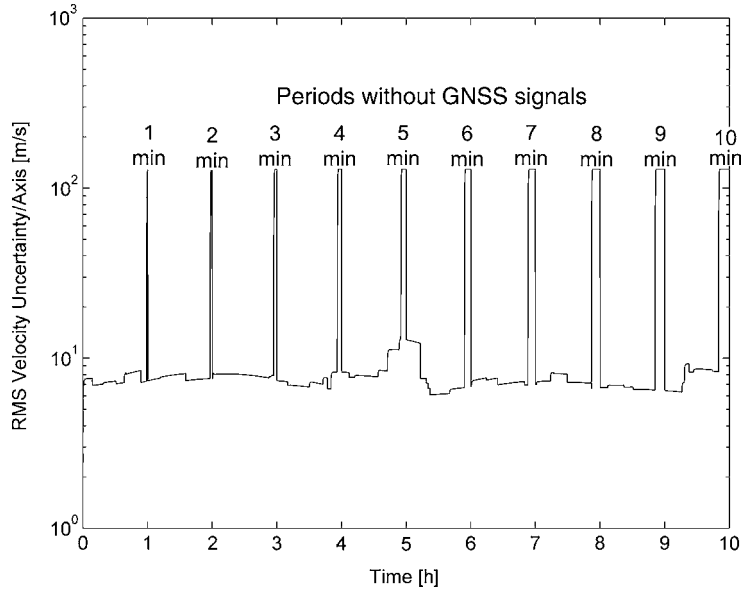


Figure 9.20 RMS velocity uncertainty for GNSS navigation on a figure-8 track.

uncertainties in this approximation would be on the order of $1^\circ - 2^\circ$. However, these vehicles would not be using the same dynamic model (MODL6) used in the track simulation, and their RMS velocity uncertainties would probably differ.

9.6 LOOSELY COUPLED GNSS/INS INTEGRATION

Evolution of Integration Methods One of the earliest and simplest implementations used GPS-derived altitude to stabilize the altitude estimates of an INS, which are otherwise unstable. Most INS implementations already included a filter for this purpose, so it was a relatively easy solution for GPS/INS integration.

This approach was soon generalized to use the horizontal components of GPS-derived position to correct horizontal components of INS position and velocity errors, as well.

Further generalizations would use GNSS measurements (including pseudoranges and pseudorange differences) and INS sensor error models for estimating and correcting variations in the sensor calibration parameters (e.g., scale factors and biases).

Loosely and Tightly Coupled Implementations GNSS/INS integration architectures have been imprecisely labeled as *loosely coupled* or *tightly coupled* according to the degree to which the Kalman filter implementation alters the internal workings of each subsystem in its stand-alone configuration.

The most loosely coupled implementations treat the standard outputs from each subsystem (i.e., GNSS or INS) as a measurement. The integrating Kalman filter combines their outputs by treating each as an independent sensor.

The more tightly coupled implementations alter the internal workings of the INS and GNSS receiver in a number of ways:

- Using nonstandard outputs of either system, such as the pseudoranges from GNSS receivers or the sensed accelerations from an INS.
- Using nonstandard inputs, such as for controlling slew rates of Doppler-tracking frequency sources in the receiver by using sensed acceleration from the INS, or altering the internal sensor compensation in the INS by using estimates of inertial sensor calibration errors derived from the integrating Kalman filter. The latter allows continuous recalibration of the INS using GNSS, so that the navigation solution is still good whenever GNSS signals are lost and signal reacquisition is faster when signals become available. Also, satellite selection can be improved by using the attitude and attitude rate information from the INS to predict which satellites will be within the GNSS receiver antenna pattern as the host vehicle maneuvers.

The main advantage of the more tightly coupled implementations is that they generally perform better than more loosely coupled implementations. One reason for this is that the Kalman filter models for tightly coupled implementations are generally more faithful representations of the actual hardware.

Cost Factors Another factor favoring more tightly coupled implementations is that their performance is generally less sensitive to the quality (and cost) of the INS. As a rule, the INS is the more expensive subsystem, so this can make a big difference in total system cost. On the other hand, the more tightly coupled implementations generally require redesign of the GNSS receiver and/or the INS to supply and/or use the additional information needed for integration. Any additional costs for such design changes must be included in the cost/benefit analysis.

9.6.1 GNSS-Aided INS: Altitude Stabilization

This is perhaps the most primitive form of GNSS/INS integration: using GNSS receiver altitude outputs to stabilize the altitude of an INS. For performance analysis, it suffices to model just the vertical component of the navigation solution, also called the *vertical channel*.

Vertical Channel Model Using Equation 9.20, we can model the dynamics of altitude error ε_h in INS as

$$\frac{d}{dt}\varepsilon_h = \dot{\varepsilon}_h \tag{9.156}$$

$$\frac{d}{dt}\dot{\varepsilon}_h = \frac{1}{\tau_S}\varepsilon_h + \varepsilon_a(t) \tag{9.157}$$

$$\frac{d}{dt}\varepsilon_a(t) = \frac{-1}{\tau_a}\varepsilon_a(t) + w(t) \tag{9.158}$$

$$\tau_S \approx 806.4 \text{ s (Schuler time constant)}, \tag{9.159}$$

where we have added an exponentially correlated random disturbance noise term $\varepsilon_a(t)$ to represent the slow drift of vertical accelerometer bias and scale factor. The correlation time constant of $\varepsilon_a(t)$ is τ_a, and the variance of the zero-mean white noise process $w(t)$ will be

$$Q_{ta} = \frac{2\,\sigma_a^2}{\tau_a}, \tag{9.160}$$

where σ_a^2 is the mean-squared accelerometer error.

The state-space model for the vertical channel in continuous time is then

$$x = \begin{bmatrix} \varepsilon_h \\ \dot{\varepsilon}_h \\ \varepsilon_a \end{bmatrix} \tag{9.161}$$

$$\dot{x} = \begin{bmatrix} 0 & 1 & 0 \\ \frac{1}{\tau_S} & 0 & 1 \\ 0 & 0 & \frac{-1}{\tau_a} \end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \\ w(t) \end{bmatrix} \tag{9.162}$$

for the zero-mean white noise process $w(t) \in \mathcal{N}(0, Q_{ta})$.

The state transition matrix for discrete time intervals Δt is then

$$\Phi = \exp(F\,\Delta t) \tag{9.163}$$

$$= \begin{bmatrix} \phi_{1,1} & \phi_{1,2} & \phi_{1,3} \\ \phi_{2,1} & \phi_{2,2} & \phi_{2,3} \\ 0 & 0 & \lambda_a \end{bmatrix} \tag{9.164}$$

$$\phi_{1,1} = \frac{\lambda_S^2 + 1}{2\,\lambda_S} \tag{9.165}$$

$$\phi_{1,2} = \frac{\tau_S(\lambda_S^2 - 1)}{2\,\lambda_S} \tag{9.166}$$

$$\phi_{1,3} = \frac{\tau_a {\tau_S}^2(\tau_S + \tau_a - 2\,\tau_a\lambda_a\lambda_S + \tau_a\lambda_S^2 - \tau_S\lambda_S^2)}{2\,\lambda_S(\tau_a^2 - \tau_S^2)} \tag{9.167}$$

$$\phi_{2,1} = \frac{(\lambda_S - 1)(\lambda_S + 1)}{2\,\tau_S\lambda_S} \tag{9.168}$$

$$\phi_{2,2} = \frac{\lambda_S^2 + 1}{\lambda_S} \tag{9.169}$$

$$\phi_{2,3} = \frac{\tau_S\tau_a(-\tau_S - \tau_a + 2\,\tau_S\lambda_a\lambda_S + \tau_a\lambda_S^2 - \tau_S\lambda_S^2)}{2\,\lambda_S(\tau_a^2 - \tau_S^2)} \tag{9.170}$$

$$\lambda_S = \exp(\Delta t/\tau_S) \tag{9.171}$$

$$\lambda_a = \exp(-\Delta t/\tau_a). \tag{9.172}$$

The corresponding process noise covariance in discrete time is

$$Q_a \approx \Delta t\, Q_{ta} \tag{9.173}$$

$$\approx \frac{2\,\Delta t\,\sigma_a^2}{\tau_a}. \tag{9.174}$$

The kth altitude outputs from the INS and GNSS receiver will be

$$\hat{h}_k^{[\mathrm{INS}]} = h_k^{[\mathrm{TRUE}]} + \varepsilon_h \tag{9.175}$$

$$\hat{h}_k^{[\mathrm{GNSS}]} = h_k^{[\mathrm{TRUE}]} + v_k, \tag{9.176}$$

respectively, where $h_k^{[\mathrm{TRUE}]}$ is the true altitude and v_k is the GNSS receiver altitude error.

The difference

$$z_k = \hat{h}_k^{[\mathrm{INS}]} - \hat{h}_k^{[\mathrm{GNSS}]} \tag{9.177}$$

$$= \varepsilon_h - v_k \tag{9.178}$$

$$= \underbrace{\begin{bmatrix} 1 & 0 & 0 \end{bmatrix}}_{H} \hat{x}_k - v_k \tag{9.179}$$

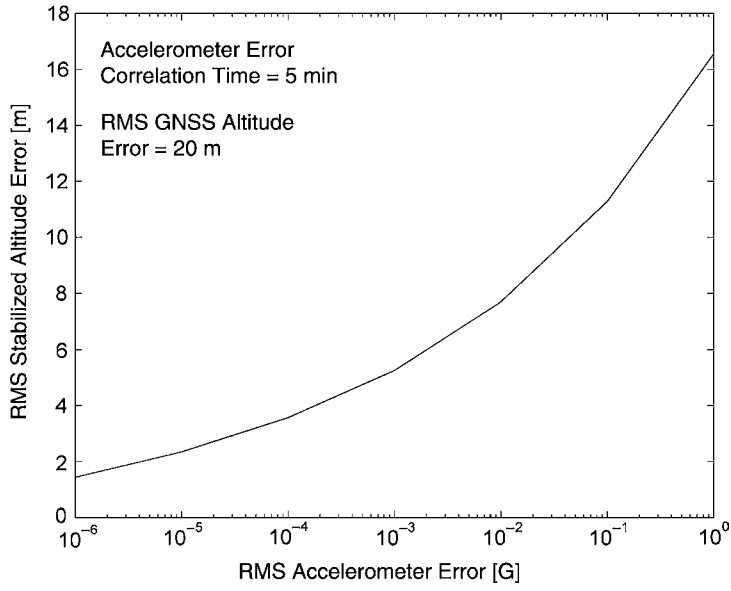


Figure 9.21 GNSS-aided INS altitude uncertainty versus RMS accelerometer error.

can then be used as a *pseudomeasurement* in a Kalman filter to estimate x. The true altitude estimate will be

$$\hat{h}_k^{[\mathrm{TRUE}]} = \hat{h}_k^{[\mathrm{INS}]} - \hat{x}_{k,\,1}, \tag{9.180}$$

where $\hat{x}_{k,1}$ is the first component of the state vector $\hat{x}_k$ of Equation 9.162.

The variance of the altitude estimate obtained in this way will be the element $p_{1,1}$ of the covariance matrix P of the Riccati equation in the Kalman filter. The steady-state solution of the continuous form of the Riccati equation can also be solved for value of $p_{1,1}$. The resulting steady-state RMS altitude uncertainty is plotted in Figure 9.21 as a function of RMS accelerometer error, for RMS GNSS altitude error equal to 20 m and its correlation time constant equal to 5 min.

The MATLAB m-file `AltStab.m` on the CD simulates and plots the time histories of RMS stabilized and "unstable" (i.e., unaided by GNSS) altitude error for the same range of RMS accelerometer noise levels as in Figure 9.21. These are Monte Carlo simulations, so the results should be different each time this m-file is run.

9.7 TIGHTLY COUPLED GNSS/INS INTEGRATION

Loss of GNSS Signals GNSS satellites use spread-spectrum signals deliberately spread over a wide frequency range. As a consequence, they have very low PSDs—so low, in fact, that they are generally below the background noise level.

GNSS signal processing relies on code despreading to pull the signals out of the noise. As a consequence, the reliability of signal detection, acquisition, and lock-on is sensitive to the SNR of the spread-spectrum signal. Receivers can easily lose satellite signals when they are partially blocked or reflected by objects overhead such as the leaves on trees. Signals can also be lost due to localized electromagnetic interference (including wideband jammers) and spoofed. High-value military targets can deploy jammers to protect themselves from GNSS-guided precision munitions.

When GNSS signals are lost, the navigation solution deteriorates rapidly, as illustrated in Figure 9.20. Kalman filter implementations help this a bit by providing velocity estimates to carry on using dead reckoning, but their accuracy falls off quickly. This has a secondary effect, because it then becomes more difficult to reacquire the satellite signals with degraded position and clock accuracies.

Benefits in INS Integration INS performance during GNSS outages can be improved by recalibrating the INS whenever GNSS is available. The approach shown here treats each subsystem (GNSS and INS) as an independent sensor, but with nonstandard outputs:

1. The GNSS receiver measures pseudoranges, which are sensitive to
   (a) True position of the vehicle antenna.
   (b) Uncompensated signal propagation delays (mostly due to the ionosphere).
   (c) Receiver clock time errors.
2. The INS measures the *vehicle state*, which includes three components each of position, velocity, acceleration, attitude, and attitude rate. However, what the INS is actually measuring is the sum of the true vehicle state plus INS navigation errors and sensor errors. Its outputs are then sensitive to
   (a) The true vehicle state, including the true position of the GNSS antenna.
   (b) INS navigation errors, which have known dynamics.
   (c) INS sensor errors, which cause navigation errors.
   All three of these are coupled together dynamically in ways that can be exploited by Kalman filtering. The filter can use the correlations caused by dynamic coupling to estimate causes (e.g., sensor errors) as well as their effects (e.g., navigation errors) by using independent measurements from GNSS. This enables the Kalman filter to recalibrate an INS in a properly integrated GNSS/INS implementation.

All these are combined into one Kalman filter representing the stochastic dynamics of all the component parts and how they interact.

Gimbaled INS Integration The next example (in Section 9.7.1) uses GNSS pseudorange measurements and INS-derived equivalent pseudoranges to estimate and compensate for INS navigation errors and sensor errors. Whenever GNSS signals are lost, this allows the INS to continue the navigation solution with its initial errors corrected.

The technical literature provides many variations of these general approaches and many alternative approaches to tightly coupled integration.

9.7.1 Gimbaled INS/GNSS Integration

This tightly coupled GNSS/INS integration approach uses GNSS pseudoranges to estimate INS navigation errors so that the INS can continue the corrected navigation solution if GNSS signals are lost. The implementation uses a nine-state INS navigation error model with errors in position, velocity, and attitude—each with three components.

This particular implementation uses the *UD*-formulated extended Kalman filter (EKF) for a full-scale example of aiding an inertial system with data provided by the GNSS receiver. For more examples and discussion, see reference [106].

The INS position output and the satellite ephemeris data can be used to compute an INS-derived pseudorange to the satellite. The difference between the INS-derived and GNSS-derived pseudoranges can be used as the measurements input to a Kalman filter to yield an integrated GNSS-aided INS. The time-differenced delta pseudoranges can also be used as measurements for estimating the relative frequency error between the transmitter and receiver clocks.

9.7.1.1 Dynamic Process Model The basic nine-state INS error model has three position errors, three velocity errors, and three tilt errors—all specified by 9×9 dynamic coefficient matrix

$$F(t) = \begin{bmatrix} -k_1 ee^{\mathrm{T}} & I_{3\times 3} & 0_{3\times 3} \\ \mathcal{A} & 2\mathcal{B} & \mathcal{C} \\ 0_{3\times 3} & 0_{3\times 3} & \mathcal{B} \end{bmatrix}, \tag{9.181}$$

$$\mathcal{A} = (3\,\omega_s^2 - k_2)ee^{\mathrm{T}} - \omega_s^2 I_{3\times 3}, \tag{9.182}$$

$$\mathcal{B} = \begin{bmatrix} 0 & \Omega & 0 \\ -\Omega & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tag{9.183}$$

$$\mathcal{C} = \begin{bmatrix} 0 & -f_3 & f_2 \\ f_3 & 0 & -f_1 \\ -f_2 & f_1 & 0 \end{bmatrix}, \tag{9.184}$$

where

e = unit vector in the vertical direction
f = specific force vector
ω_s = Schuler frequency

TABLE 9.5 Inertial Sensor Error Sources (1σ)

Accelerometer	
G-insensitive	
Bias stability	40 μG
Scale factor stability	100 ppm
Scale factor asymmetry	100 ppm
Nonorthogonality	$1.1_{10}-6$ arc-sec
White noise	5 μG/Hz$^{1/2}$
Correlated noise	4 μG
Correlation time	20 min
G-sensitive	
Nonlinearity	5 μG/G^2
Cross-axis coupling	5 μG/G^2
Gyroscope	
G-insensitive	
Bias stability	0.001 deg/h
Scale factor stability	100 ppm
Scale factor asymmetry	100 ppm
Nonorthogonality	$1.1_{10}-6$ arc-sec
White noise	0.002 deg/h/Hz$^{1/2}$
Correlated noise	0.004 deg/h
Correlation time	20 min
G-sensitive	
Mass unbalance	0.008 deg/h/G
Quadrature	0.008 deg/h/G
Anisoelastic	0.001 deg/h/G^2

Ω = earth spin rate
k_1 = vertical-channel position loop gain
k_2 = vertical-channel velocity loop gain

and values shown in Tables 9.5.

9.7.1.2 Measurement Model As given in reference [106, 149], the GNSS pseudorange (ρ) from a satellite is defined as

$$\rho = [(X_S - X_R)^2 + (Y_S - Y_R)^2 + (Z_S - Z_R)^2]^{1/2} + c\,C_b, \tag{9.185}$$

where

(X_S, Y_S, Z_S) = satellite position coordinates at the time of transmission
(X_R, Y_R, Z_R) = receiver position coordinates at the time of reception
C_b = receiver clock bias error
c = carrier speed (speed of light)

The linearized observation equation implemented in the EKF is

$$\delta\rho = H_\rho \delta x + V_\rho, \tag{9.186}$$

where δx is the state vector error with its states (three position errors, three velocity errors, and three platform tilt errors); V_ρ is the additive measurement noise; and H_ρ, the pseudorange observation matrix, is obtained by linearizing the pseudorange equation with respect to the filter states (Jacobian matrix):

$$H_\rho = \begin{bmatrix} \frac{\partial\rho_1}{\partial x} & \frac{\partial\rho_1}{\partial y} & \frac{\partial\rho_1}{\partial z} & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{\partial\rho_2}{\partial x} & \frac{\partial\rho_2}{\partial y} & \frac{\partial\rho_2}{\partial z} & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{\partial\rho_3}{\partial x} & \frac{\partial\rho_3}{\partial y} & \frac{\partial\rho_3}{\partial z} & 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial\rho_j}{\partial x} & \frac{\partial\rho_j}{\partial y} & \frac{\partial\rho_j}{\partial z} & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{9.187}$$

$$\left.\frac{\partial\rho_j}{\partial X}\right|_{X=\hat{X}} = \frac{\hat{x}R - x_j}{\hat{\rho}_j} \tag{9.188}$$

$$\left.\frac{\partial\rho_j}{\partial Y}\right|_{Y=\hat{Y}} = \frac{\hat{y}R - y_j}{\hat{\rho}_j} \tag{9.189}$$

$$\left.\frac{\partial\rho_j}{\partial Z}\right|_{Z=\hat{Z}} = \frac{\hat{z}R - z_j}{\hat{\rho}_j}. \tag{9.190}$$

$j = 1, 2, 3, \ldots$ is satellite number.

The GNSS delta pseudorange is defined as the difference between two pseudoranges separated in time:

$$\Delta\rho = \rho_k - \rho_{k-1}. \tag{9.191}$$

Since the delta pseudorange represents the Doppler integrated over a finite time interval, any point within the integration interval can be chosen as the reference time at which the measurement is valid. In practice, either the beginning or end of the interval is selected as the reference time.

If the interval ending time is chosen as the reference, the linearized measurement model can be written as

$$\Delta\rho = H_{\Delta\rho}\delta X + v_{\Delta\rho}, \tag{9.192}$$

where the measurement noise $v_{\Delta\rho}$ not only accounts for the very small additive tracking error in the highly accurate carrier loop but also includes the integrated dynamic

effects representing unmodeled vehicle jerk variations and higher order terms over the integration interval, and

$$H_{\Delta\rho} = \left.\frac{\partial \Delta\rho}{\partial X}\right|_{X=\hat{X}} \tag{9.193}$$

$$= \begin{bmatrix} 0 & 0 & 0 & h_x^1 & h_y^1 & h_z^1 & 0 & 0 & 0 \\ 0 & 0 & 0 & h_x^2 & h_y^2 & h_z^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & h_x^3 & h_y^3 & h_z^3 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & h_x^j & h_y^j & h_z^j & 0 & 0 & 0 \end{bmatrix}. \tag{9.194}$$

9.7.1.3 Kalman Filter State Configuration Figure 9.22 shows a block diagram representation of an INS using inertial and satellite information. The integrated GNSS/inertial system provides the estimated position and velocity during GNSS signal availability and extends the period of acceptable operation subsequent to the loss of GNSS signals. As proposed by Bletzacker et al. [40], an important part of the filter design process involves the selection of a state configuration that can satisfy the performance requirements within existing throughput constraints. Other than the basic nine states (position, velocity, and platform error angles) required in any aided mechanization, the remaining states are chosen from the suite of inertial sensor error parameters. The effect that these parameters have on system performance depends critically upon whether the INS is gimbaled or strapdown. For example, errors in gyroscope scale factor, scale factor asymmetry, and nonorthogonality will be more important in a strapdown system, where the gyroscopes are exposed to a much higher level of dynamics and their effect on system performance is reduced significantly. For this reason, all of the inertial sensor errors considered for inclusion as states, except for gyroscope drift, are related to the accelerometers. The selection process included a

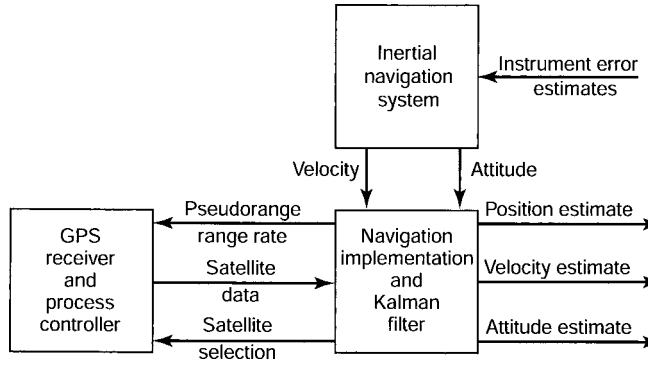


Figure 9.22 Integrated GNSS/INS navigation system.

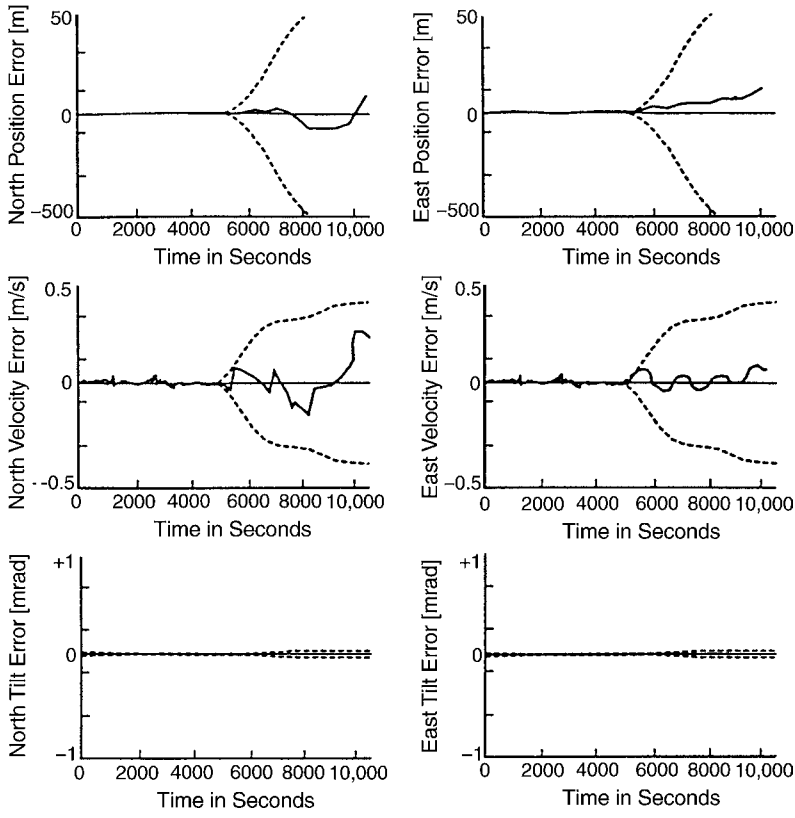


Figure 9.23 Integrated GNSS/INS simulation results.

trade-off study involving options ranging from 9 (position, velocity, and platform tilts) to 24 (position, velocity, platform tilts, accelerometer bias, gyroscope drifts, accelerometer scale factor and scale factor asymmetry, and accelerometer nonorthogonality) states. The ultimate decision was based upon considerations of throughput, performance requirements, sensor characteristics, and mission applications. The result of this study was the selection of a 15-state Kalman filter with states of position, velocity, platform error angles, gyroscope drift, and accelerometer bias [40].[10]

The modeled INS is a 0.5-nautical mile/hour (CEP rate) system. The INS vertical channel is assumed to be stabilized by an error-free barometric altimeter. The RMS GNSS pseudorange and delta pseudorange measurement errors are 0.6 m and 2.0 cm, respectively. The INS vertical channel position, velocity, and acceleration control loop gains are 0.03, 0.0003, and 0.000001, respectively [283]. The GNSS control and space segments are assumed to have bias errors of 5 m. The GNSS receiver

[10]Other investigators have evaluated filters with 39, 12, and 14 states. Maybeck [184] has mentioned a 96-state error state vector. In this example, we give the results of a 15-state filter.

clock has no G-sensitivity. All lever-arm effects (offset of the GNSS antenna from the INS center) have been omitted. An 18-satellite constellation is assumed to be operational, with 3 or 4 continuously available. The flight profile includes a take-off and climb to 7 km with an acceleration (5 m/s/s) to a speed of 300 m/s. The aircraft then flies a racetrack loop with 180-km straight legs and 7-m/s/s turns [40]. The GNSS is assumed to be available for the first 5000 s.

Table 9.5 gives typical error source characteristics for this application. A typical set of results is shown in Figure 9.23. The error growth in the receiver position and velocity estimates is caused by the inertial reference tilt errors while GNSS data are lost (after 5000 s). The improved tilt estimation provided by the Kalman filter implementation may provide an order-of-magnitude improvement in the resulting (integrated) navigation solution.

9.7.2 Strapdown INS/GNSS Integration

The fact that strapdown sensor axis directions are unconstrained makes the modeling for GNSS/INS integration a bit more complicated. The models given here are exact, except for some aspects of attitude errors. We use a model based on Euler angles (roll, pitch, and yaw), which will not work when the vehicle roll axis is pointed straight up. The simulation conditions avoid this by limiting the range of roll, pitch, and heading angles such that some small-angle approximations can be used. The resulting models may be adequate for performance analysis with the Riccati equation but not for implementing the full Kalman filter. The full model has 67 state variables, although 18–20 of them (corresponding to uncompensated GNSS satellite signal propagation delays) are not being estimated at all times.

State Vector The model state vector in this example includes subvectors for the true vehicle state, clock errors, GNSS signal propagation delay errors, INS navigation errors, and sensor errors. The complete state vector is composed of subvectors for these five submodels:

$$x = \begin{bmatrix} x_{\text{vehicle}} \\ x_{\text{delay}} \\ x_{\text{clock}} \\ x_{\text{nav}} \\ x_{\text{sens}} \end{bmatrix}, \tag{9.195}$$

where

x_{vehicle} is the vehicle state vector, the dynamics of which are statistically constrained by stochastic dynamic model number 6 in Table 9.1 with parameters determined by track statistics in Tables 9.3 and 9.6. It has 15 state variables.

x_{delay} is the vector of satellite propagation delay errors, statistically constrained by the stochastic dynamic model of Section 9.5.2. It has one state variable for each satellite being tracked. In MATLAB satellite simulators in the CD, 29 satellite

TABLE 9.6 Vehicle Attitude Statistics from Figure-8 Track Simulator

State Variable		Mean	Std. Dev.		Corr.
Name	Symbol	μ	σ	Units	Time [s]
North tilt	(θ_N)	0.0000	0.1690	[rad]	25
East tilt	(θ_E)	0.0000	0.1756	[rad]	6
Heading	(θ_D)	−1.5708	1.5357	[rad]	22
North rate	$(\dot{\theta}_N)$	0.0197	0.0112	[rad/s]	10
East rate	$(\dot{\theta}_E)$	−0.0015	0.0117	[rad/s]	17
Heading rate	$(\dot{\theta}_D)$	0.0000	0.1647	[rad/s]	14

signal delay state variables may be implemented in the models, although only 9–11 satellites may be usable at any one time.

x_{clock} is the clock error state vector, statistically constrained by the stochastic dynamic model of Equation 9.5.1. It has two state variables.

x_{nav} is the INS navigation error state vector, statistically constrained by the stochastic dynamic model of Sections 9.3.5.7 and 9.3.5.8. It has nine state variables.

x_{sens} is a vector of sensor errors, statistically constrained by the stochastic dynamic models of Section 9.3.6. It has 12 state variables.

The total number of possible state variables in this model is 67, and this is the number used in the simulations. The minimum number would be 47–49 if only those propagation delays for the satellites being tracked were used.

GNSS pseudoranges measure only the three position components of x_{vehicle}, and 9–11 such pseudoranges can be available at any time in the simulations.

The INS outputs measure the sum of the true vehicle state (15 state variables) and the INS navigation errors (15 of the 21 INS state variables).

The Kalman filter uses both types of measurements (pseudoranges and INS outputs) to estimate the values of all the variables. This reduces the uncertainty in the estimates of all state variables over what either INS or GNSS could achieve alone.

It is the improvement in the estimates of the true vehicle state, the INS navigation errors, and INS sensor errors that improves robustness against loss of GNSS signals. When GNSS signals are lost, the reduced uncertainty in the values of the true vehicle state, the INS navigation errors, and the INS sensor errors improves the short-term navigation accuracy over what could be accomplished with GNSS alone.

9.7.2.1 Host Vehicle State Vector The application is for tracking a car on a figure-8 test track using GNSS pseudoranges, which are only sensitive to vehicle position. The vehicle model used in Section 9.5.5.2 for this purpose has the first 9 of the following 15 state variables:

P_N = northing from center of track

V_N = north velocity

A_N = north acceleration
P_E = easting from center of track
V_E = east velocity
A_E = east acceleration
P_D = downward displacement from center of track
V_D = downward velocity
A_D = downward acceleration
θ_N = tilt about north axis
$\dot{\theta}_N$ = rotation rate about north axis
θ_E = tilt about east axis
$\dot{\theta}_E$ = rotation rate about east axis
θ_D = heading angle
$\dot{\theta}_D$ = heading angle rate

The first nine elements of the vehicle state vector use dynamic model number 6 (bounded RMS position) from Table 9.1 for each axis of motion, with dynamic disturbance noise covariance determined from the empirical statistical parameters in Table 9.3. The last six state variables are added for the GNSS/INS integration model.

Attitude and Attitude Rate Attitude and attitude rate are not part of the vehicle state vector used in GNSS simulations, because they are unobservable from GNSS measurements. However, the statistics of attitude and attitude rate from the track simulator in Table 9.6 can be used to derive a simple statistical model for attitude dynamics.

Heading angle has a rather large mean and standard deviation. Due to track geometry and the direction of travel, the vehicle heading never comes within about $\pm 56.3^\circ$ of due east.

These statistics are a reasonable fit to a critically damped dynamic system model,

$$\frac{d}{dt}\begin{bmatrix} \theta - \mu_\theta \\ \dot{\theta} - \mu_{\dot{\theta}} \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 \\ -\tau^{-2} & -2\,\tau^{-1} \end{bmatrix}}_{F} \begin{bmatrix} \theta - \mu_\theta \\ \dot{\theta} - \mu_{\dot{\theta}} \end{bmatrix} + \begin{bmatrix} 0 \\ w_{\ddot{\theta}}(t) \end{bmatrix}, \tag{9.196}$$

with the steady-state Riccati equation solution

$$0 = FP + PF^T + Q \tag{9.197}$$

$$Q = \begin{bmatrix} 0 & 0 \\ 0 & \sigma_{\ddot{\theta}}^2 \end{bmatrix} \tag{9.198}$$

$$p_{1,2} = 0 \tag{9.199}$$

$$p_{1,1} = \sigma_{\ddot{\theta}}^2 \tau^3 / 4 \tag{9.200}$$

$$\stackrel{\text{def}}{=}\sigma_\theta^2 \text{ (variance of } \theta \text{ about } \mu_\theta)$$

$$p_{2,2} = \sigma_{\dot{\theta}}^2 \tau/4 \tag{9.201}$$

$$\stackrel{\text{def}}{=}\sigma_{\dot{\theta}}^2 \text{(variance of } \dot{\theta} \text{ about } \mu_{\dot{\theta}})$$

$$\sigma_{\dot{\theta}}^2 = \frac{4\,\sigma_\theta^2}{\tau^3} \tag{9.202}$$

$$= \frac{4\,\sigma_{\dot{\theta}}^2}{\tau}. \tag{9.203}$$

For correlation time τ on the order of 10 s, this makes the standard deviation of $\sigma_{\dot{\theta}}$ about an order of magnitude less than that of σ_θ. This is very close to the statistics in Table 9.6.

The analogous discrete-time dynamic model is

$$\Phi_\theta = \exp(\Delta t\, F) \text{ (state transition matrix)} \tag{9.204}$$

$$= e^{-\Delta t/\tau}\begin{bmatrix} (\tau + \Delta t)/\tau & \Delta t \\ -\Delta t\,/\tau^2 & (\tau - \Delta t)/\tau \end{bmatrix} \tag{9.205}$$

$$\begin{bmatrix} \hat{\theta}_k \\ \hat{\dot{\theta}}_k \end{bmatrix} = \Phi_\theta \begin{bmatrix} \hat{\theta}_{k-1} - \mu_\theta \\ \hat{\dot{\theta}}_{k-1} - \mu_{\dot{\theta}} \end{bmatrix} + \begin{bmatrix} \mu_\theta \\ \mu_{\dot{\theta}} \end{bmatrix} \tag{9.206}$$

$$P_k = \Phi_\theta P_{k-1} \Phi_\theta^T + Q_\theta \tag{9.207}$$

$$Q_\theta \approx \begin{bmatrix} 0 & 0 \\ 0 & \sigma_{\dot{\theta}}^2\,\Delta t \end{bmatrix}. \tag{9.208}$$

This simplified model was used for the vehicle attitude angles and rates during simulated performance analysis, with parameters determined from Table 9.6. The MATLAB function `CDModelParams.m` on the CD computes the dynamic model parameters Φ_θ and Q_θ, given the angular variances and correlation times from Table 9.6. These models are for performance analysis only. A more rigorous model would be required for estimating attitude in the integrated GNSS/INS Kalman filter.

9.7.2.2 Strapdown Attitude Model

RPY Coordinates This model assumes a strapdown INS with sensor axes nominally aligned with the host vehicle roll, pitch, and yaw axes, as illustrated in Figure 9.24. Here, the roll, pitch, and yaw axes are labeled R, P, and Y, respectively. The north, east, and downward axes of a locally level coordinate system are labeled N, E, and D, respectively, and the roll, pitch and yaw (heading) angles are labeled θ_R, θ_P, and θ_Y, respectively.

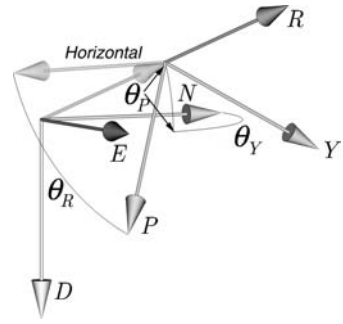


Figure 9.24 RPY and NED coordinates.

Roll, *pitch*, and *yaw* were originally nautical terms used for describing the gyrations of ships in heavy seas. They generally refer to vehicle-fixed axes and rotations about these axes.

The vehicle roll axis is nominally pointed forward, out the "front" of the vehicle and along the longitudinal axis of a ship or rocket.

The vehicle pitch axis is nominally out the right-hand side of the vehicle, so that elevating the roll axis (nose) of the vehicle above the horizontal corresponds to a positive pitch angle. The pitch axis will be horizontal when the vehicle roll angle is zero.

The vehicle yaw axis completes a right-handed coordinate system, so that it points downward when the vehicle roll and pitch angles are zero. When

$$\theta_R = \theta_P = \theta_Y = 0,$$

RPY coordinates are aligned with NED coordinates.

Coordinate Transformation The coordinate transform from RPY coordinates to NED coordinates can be represented in terms of the vehicle roll, pitch, and yaw/heading angles as

$$C_{NED}^{RPY} = C_Y C_P C_R \tag{9.209}$$

$$C_R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta_R) & -\sin(\theta_R) \\ 0 & \sin(\theta_R) & \cos(\theta_R) \end{bmatrix} \tag{9.210}$$

$$C_P = \begin{bmatrix} \cos(\theta_P) & 0 & \sin(\theta_P) \\ 0 & 1 & 0 \\ -\sin(\theta_P) & 0 & \cos(\theta_P) \end{bmatrix} \tag{9.211}$$

$$C_Y = \begin{bmatrix} \cos(\theta_Y) & -\sin(\theta_Y) & 0 \\ \sin(\theta_Y) & \cos(\theta_Y) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{9.212}$$

where

θ_R = vehicle roll angle
θ_P = vehicle pitch angle
θ_Y = vehicle yaw angle

9.7.2.3 INS Error Model The state variables of the INS error model include the INS navigation errors ε_N, ε_E, ε_D, $\dot{\varepsilon}_N$, $\dot{\varepsilon}_E$, and $\dot{\varepsilon}_D$ from Equation 9.12, which are the errors in the INS-estimated P_N, P_E, P_D, V_N, V_E, and V_D, respectively. The errors $\ddot{\varepsilon}_N$, $\ddot{\varepsilon}_E$, and $\ddot{\varepsilon}_D$ in the INS estimates of A_N, A_E, and A_N are included in the sensor error model.

The dynamic coefficient matrix for navigation errors is given in Equation 9.61 and computed by the MATLAB function `FNNmat.m` on the CD.

9.7.2.4 INS Sensor Error Model The sensor error state vector is a 12-state subvector of that in Equation 9.77, including only bias and scale factor errors:

$$x_S^T = [\varepsilon_{\text{abR}},\ \varepsilon_{\text{abP}},\ \varepsilon_{\text{abY}},\ \varepsilon_{\text{asR}},\ \varepsilon_{\text{asP}},\ \varepsilon_{\text{asY}},\ \varepsilon_{\text{gbR}},\ \varepsilon_{\text{gbP}},\ \varepsilon_{\text{gbY}},\ \varepsilon_{\text{gsR}},\ \varepsilon_{\text{gsP}},\ \varepsilon_{\text{gsY}}], \tag{9.213}$$

in ISA-fixed RPY coordinates.

Sensor Error Dynamics The dynamic coefficient matrix F_{SS} for the sensor error model is a 12×12 submatrix of that given in Equation 9.78:

$$F_{SS} = \begin{bmatrix} -\tau_{abR}^{-1} & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & -\tau_{abP}^{-1} & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & -\tau_{abY}^{-1} & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\tau_{asR}^{-1} & \cdots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & -\tau_{gbY}^{-1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & -\tau_{gsR}^{-1} & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & -\tau_{gsP}^{-1} & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & -\tau_{gsY}^{-1} \end{bmatrix}, \tag{9.214}$$

and its 12×12 covariance matrix of dynamic disturbance noise is

$$Q_{SS} = \begin{bmatrix} 2\sigma_{abR}^2/\tau_{abR} & 0 & 0 & \cdots & 0 \\ 0 & 2\sigma_{abP}^2/\tau_{abP} & 0 & \cdots & 0 \\ 0 & 0 & 2\sigma_{abY}^2/\tau_{abY} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 2\sigma_{gsY}^2/\tau_{gsY} \end{bmatrix}, \tag{9.215}$$

where σ^2_{abR} is the mean-squared bias error of the roll accelerometer and τ_{abR} is its correlation time constant, The other parameters of Equation 9.215 are defined similarly for the accelerometer and gyroscope bias errors and scale factor errors.

Both F_{SS} and Q_{SS} are computed by the MATLAB function `FSSQSmat` on the CD.

9.7.2.5 INS Measurement Model

INS Measurement Vector The INS measures the sum of true vehicle state and INS errors, which include navigation errors and sensor errors. Its modeled measurement vector then has 15 variables:

$$z_{INS} = \begin{bmatrix} P_N + \varepsilon_N \\ P_E + \varepsilon_E \\ P_D + \varepsilon_D \\ V_N + \dot{\varepsilon}_N \\ V_E + \dot{\varepsilon}_E \\ V_D + \dot{\varepsilon}_D \\ \begin{bmatrix} A_N \\ A_E \\ A_D \end{bmatrix} + C^{RPY}_{NED} \left\{ \begin{bmatrix} \varepsilon_{abR} \\ \varepsilon_{abP} \\ \varepsilon_{abY} \end{bmatrix} + \begin{bmatrix} a_R & 0 & 0 \\ 0 & a_P & 0 \\ 0 & 0 & a_Y \end{bmatrix} \begin{bmatrix} \varepsilon_{asR} \\ \varepsilon_{asP} \\ \varepsilon_{asY} \end{bmatrix} \right\} \\ \theta_N + \varepsilon_{\theta N} \\ \theta_E + \varepsilon_{\theta E} \\ \theta_D + \varepsilon_{\theta D} \\ \begin{bmatrix} \omega_N \\ \omega_E \\ \omega_D \end{bmatrix} + C^{RPY}_{NED} \left\{ \begin{bmatrix} \varepsilon_{gbR} \\ \varepsilon_{gbP} \\ \varepsilon_{gbY} \end{bmatrix} + \begin{bmatrix} \omega_R & 0 & 0 \\ 0 & \omega_P & 0 \\ 0 & 0 & \omega_Y \end{bmatrix} \begin{bmatrix} \varepsilon_{gsR} \\ \varepsilon_{gsP} \\ \varepsilon_{gsY} \end{bmatrix} \right\} \end{bmatrix}. \quad (9.216)$$

Acceleration Measurement Sensitivity to Sensor Errors The measurement sensitivity matrix for this INS model for a *navigation sensor* is computed by the MATLAB function `HINS67`. Measurements also include GNSS pseudoranges. The sensitivity of INS-estimated accelerations in NED coordinates to these variables is

$$\frac{\partial A_{NED}}{\partial \varepsilon_S} = C^{RPY}_{NED} \begin{bmatrix} 1 & 0 & 0 & a_R & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & a_P & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & a_Y & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad (9.217)$$

where a_R, a_P, and a_Y are sensed accelerations in RPY coordinates.

9.7.2.6 Simulation Results Several MATLAB scripts were used for simulating different navigation configurations:

`Fig8FreeInert.m` simulates navigation performance for an INS without aiding (except for vertical channel stabilization).

`Fig8HVINS.m` simulates navigation performance for an INS without sensor aiding (except for vertical channel stabilization), but using a host vehicle dynamic model to statistically constrain navigation errors. This host vehicle dynamic model is "tuned" to the statistical dynamic conditions on the figure-8 test track. The same host vehicle model is used in the following simulators as well.

`Fig8GNSSonly.m` uses only GNSS navigation.

`Fig8GNSSINS.m` calculates and plots RMS performance statistics of a tightly coupled strapdown INS/GNSS integrated system which estimates and compensates for INS navigation errors and sensor errors.

All simulations were for the same dynamic conditions (the figure-8 track) and the same satellite geometries.

Benefits of Host Vehicle Modeling Figure 9.25 is a plot of INS performance with and without using a host vehicle model to constrain navigation errors. Performance here is in terms of RMS horizontal postion uncertainty. The improvement in this case is rather extreme, which may be due to unusual confinement of the simulated trajectory and/or the high quality of the INS.

Benefits of Integration Figure 9.26 is a plot of the performance of

1. Navigation with GNSS by itself.
2. Navigation with an INS by itself.
3. Navigation with a tightly coupled integrated strapdown INS/GNSS system.

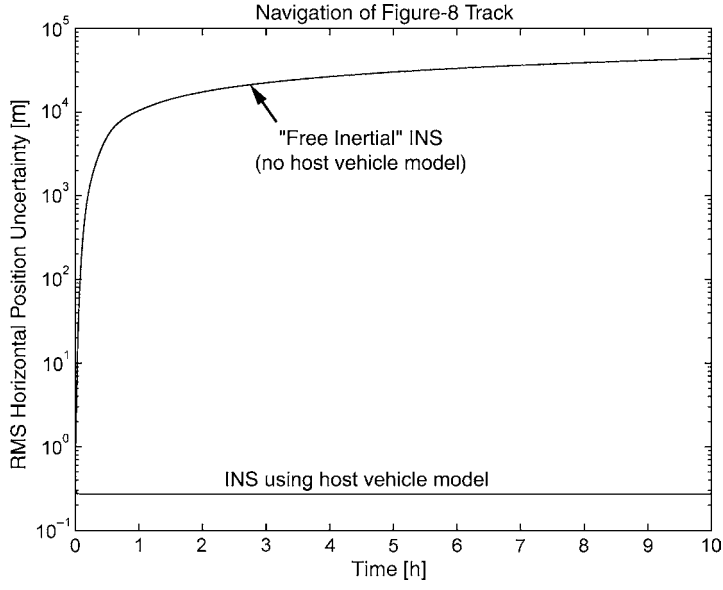


Figure 9.25 RMS horizontal position uncertainties for INS navigation.

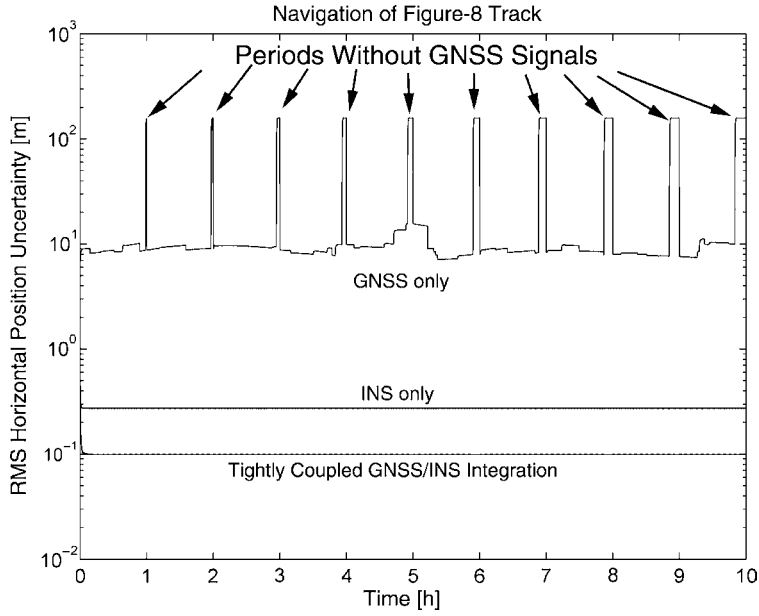


Figure 9.26 Relative performance of GNSS, INS, and integrated GNSS/INS.

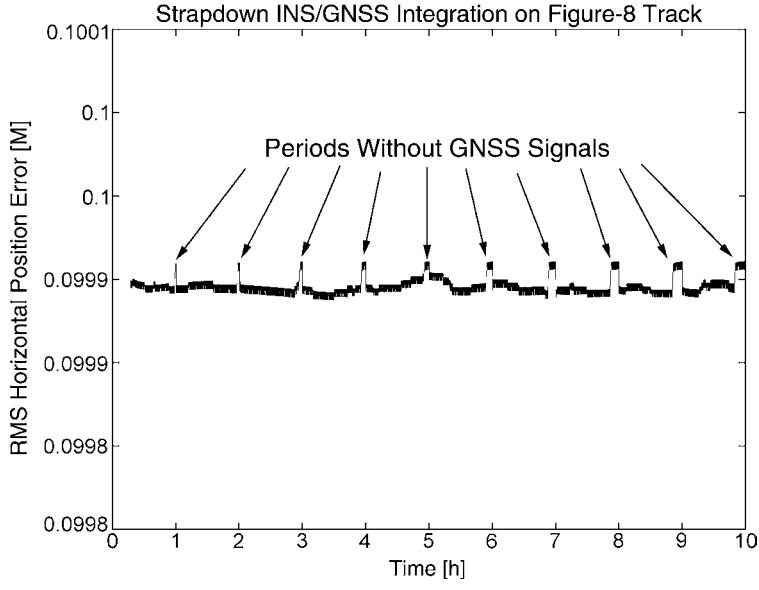


Figure 9.27 Effect of GNSS signal loss on integrated performance.

All simulations are for 10 h, with GNSS signals unavailable for the last n minutes of the nth hour.

At the scales used in the plot, the effects of GNSS signal loss on integrated GNSS/INS navigation performance are not evident. The effect is so small (perhaps due to the high quality of the INS) that the data had to be filtered to bring it out. The corrupting harmonics have periods of 30 and 60 s, which are the fundamental frequencies of the figure-8 track dynamics. The result is plotted in Figure 9.27, showing a tiny degradation when GNSS signals are lost for ≤ 10 min.

The real power of GNSS/INS integration is that it radically relaxes the performance requirements on the INS as a stand-alone sensor to achieve a given level of integrated navigation performance.

9.7.3 INS-Aided GNSS Satellite Signal Phase Tracking

Satellite Signal Tracking GNSS receivers need to correct for satellite signal Doppler shifts due to relatively high satellite velocities.[11]

Receiver Tracking Loops In GNSS stand-alone receivers, frequency-lock loops and phase-lock loops are used to maintain signal lock-on. These types of control loops have limited dynamic capabilities, and they can easily loose signal-lock during violent maneuvers of high-g vehicles such as missiles or unmanned autonomous vehicles.

INS-Aided Signal Tracking In integrated GNSS/INS implementations, accelerations sensed by the INS can be used as an aid in maintaining signal lock-on during such maneuvers. This type of GNSS receiver aiding can significantly improve signal lock-on during such maneuvers.

GNSS Receiver Antenna Switching Because attitude maneuvers of the host vehicle could place the direction to a satellite outside the antenna pattern of a GNSS antenna mounted on the surface of a host vehicle, highly maneuverable vehicles generally require more than one antenna to maintain signal lock-on. In that case, attitude and attitude rate information from the INS can also aid the antenna switching implementation.

9.8 SUMMARY

9.8.1 Important Points to Remember:

1. Kalman filters have had an enormous impact on the accuracy of navigation, and many new navigation systems (e.g., GNSS and integrated GNSS/INS) have been designed to use Kalman filters.

[11] A few GNSS satellites may be in geostationary orbits. In order to maintain global coverage, however, most satellites will be in lower orbits with relatively high inclination angles to the equator.

2. When a navigation system is being designed for a specific type of host vehicle (e.g., cargo ship, automobile, fighter aircraft), the Kalman filter design can improve navigation performance by using a dynamic model tailored to the maneuvering capabilities of the host vehicle.
3. The design possibilities for integrating satellite navigation with inertial navigation can be limited by the available inputs and outputs of the GNSS receiver and INS, and by the extent to which the integration designer can alter the inner workings of each subsystem (GNSS receiver or INS). This has led to a variety of integration approaches:
   (a) *Loosely coupled* implementations are more conservative in terms of how much the integration mechanization alters the standard inputs and outputs of each subsystem.
   (b) *Tightly coupled* implementations can alter the more intimate details of the subsystem implementations and generally can perform better than loosely coupled implementations.
4. We have provided realistic examples of linear stochastic system models for GNSS and INS error dynamics, and have shown how these models behave under representative dynamic conditions.
5. The Riccati equation of the Kalman filter is an indispensable tool for the design of navigation systems, especially those integrating sensors with very different error characteristics such as GNSS and INS. The Riccati equation allows us to predict the performance of a particular system design as a function of subsystem and component error characteristics. Designers can then search for the combination of components to satisfy a specified set of performance characteristics at minimum system cost.
6. Medium-accuracy intertidal navigators (1 nautical mile per hour CEP rate) cost about $\$US10^5$. GNSS receivers cost about $\$US10^2$. Therefore, the INS is the "long pole in the tent" for integrated system cost.
7. The performance of integrated GNSS/INS navigation systems is not very sensitive to the stand-alone performance of the INS, which means that very-low-cost INS technologies can be used to bring down the total integrated system cost. This technology development is currently driving down the cost for high-accuracy integrated GNSS/INS systems for such widespread applications as automated farm equipment control, automated grading for roads, etc. The potential monetary payoff for this capability is substantial.
8. Kalman filtering will play an important part in this development.

PROBLEMS

9.1 Prove the results shown for H_ρ in Equation 9.187 and for $H_{\Delta\rho}$ in Equation 9.194.

9.2 Formulate the GPS plant model with three position errors, three velocity errors, and three acceleration errors and the corresponding measurement model with pseudorange and delta pseudorange as measurements.

9.3 Run the MATLAB program `Example0901.m` with clock stability values of 10^{-9}, 10^{-8}, 10^{-7}, and 10^{-6}. This is for the best conditions: a stationary receiver with good satellite geometry. How is the estimate of clock drift C_d affected by clock stability under the best conditions?

9.4 Run the MATLAB program `Example0906.m` with the following two sets of satellite directions:

Set	Satellite	Elevation	Azimuth
1	1	0	0
	2	0	120
	3	0	240
	4	90	0
2	1	45	0
	2	45	90
	3	45	180
	4	45	270

Which has better performance? Why?

9.5 Run the MATLAB m-file `AltStab.m` for simulating GNSS-aided altitude stabilization. Follow the instructions displayed at the end to display the roots of the tenth-order polynomial equation for the steady-state mean-squared altitude estimation uncertainty as a function of RMS accelerometer noise. Successive columns represent 10^{-6}, 10^{-5}, 10^{-6}, . . . , 10^{0} g RMS accelerometer noise, and the rows are the polynomial roots. Can you guess which roots are the correct values? If so, plot their square roots (i.e., the steady-state RMS altitude uncertainty) versus the RMS accelerometer noise σ_a. How do these results compare to the final plot displayed?

9.6 Add clock errors to the model in `GNSSshootoutNCE.m`. Assume 10^{-9} relative clock stability over 1 s. (See how it is done in Example 9.5.) Compare your results with those in Table 9.4.

9.7 Solve Equation 9.9 for P as a function of the elements of F and Q.

9.8 In Figure 9.26, RMS position error is bounded whenever GNSS signals are lost, whether for 1 min or 10 min. Why?

APPENDIX A

MATLAB SOFTWARE

The accompanying CD contains MATLAB functions and scripts for implementing the Kalman filter and demonstrating its use. The ASCII file `READMEFIRST.TXT` in the root directory should be read before starting to use any of the software. It describes the current contents and directory structure of the files on the CD.

A.1 NOTICE

This software is intended for demonstration and instructional purposes only. The authors and publishers make no warranty of any kind, expressed or implied, that these routines meet any standards of mercantibility for commercial purposes. These routines should not be used as-is for any purpose or application that may result in loss or injury, and the publishers and authors shall not be liable in any event for incidental or consequential damages in connection with or arising out of the furnishing, performance, or use of these programs.

A.2 GENERAL SYSTEM REQUIREMENTS

The MATLAB scripts on the CD are designed for MATLAB environments. Information on MATLAB can be obtained from

Kalman Filtering: Theory and Practice using MATLAB®, Third Edition. By Mohinder S. Grewal and Angus P. Andrews

The Math Works, Inc.
3 Apple Hill Drive
Natick, MA 01760 USA

Tel: 508-647-7000
Fax: 508-647-7101
E-mail: info@mathworks.com
Web: www.mathworks.com

You may also use the MATLAB editor to modify these scripts as needed. Comments in the listings contain additional information on the calling sequences and array dimensions.

A.3 CD DIRECTORY STRUCTURE

The CD directories are organized by the chapters in which the supporting concepts are presented. The ASCII file `WHATSUP.DOC` in the root directory describes any changes made in the directory structure or software after printing. It should also be read before starting to use any of the software.

There are some common functions that are not repeated in every directory containing a calling script, so users are encouraged to copy all MATLAB scripts to a common subdirectory of the MATLAB "work" directory on their own computers.

A.4 MATLAB SOFTWARE FOR CHAPTER 2

The directory CHAPTER2 contains `expm1.m`, referred to in Chapter 2 and Appendix B Section B.4.4.2.

A.5 MATLAB SOFTWARE FOR CHAPTER 3

`VanLoan.m` implements Van Loan's method for computing Kalman filter model parameter matrices (i.e., in discrete time) from Kalman–Bucy filter model parameter matrices (i.e., in continuous time).

A.6 MATLAB SOFTWARE FOR CHAPTER 4

The directory CHAPTER4 contains software implementing the algorithms defined in Chapter 4. See the file `WHATSUP.DOC` in the root directory for descriptions of any changes.

`demo1.m` is a MATLAB script for demonstrating the effects of process noise and observations of the probability distribution of a single state variable as a function of time. The plots show the evolution of the probability density function while observations are made at discrete times.

`exam43.m` is a MATLAB script for demonstrating Example 4.3 in MATLAB.

`exam44.m` is a MATLAB script for demonstrating Example 4.4 in MATLAB.

`obsup.m` is a MATLAB script implementation of the Kalman filter observational update, including the state update and the covariance update (Riccati equation).

`timeup.m` is a MATLAB script implementation of the Kalman filter temporal update, including the state update and the covariance update (Riccati equation).

A.7 MATLAB SOFTWARE FOR CHAPTER 5

`sim3pass.m` generates a trajectory for a scalar linear time-invariant model driven by simulated noise for measurement noise and dynamic disturbance noise, applies a three-pass fixed-interval smoother to the resulting measurement sequence, and plots the resulting estimates along with the true (simulated) values and computed variances of estimation uncertainty.

`FiniteFIS.m` computes and plots the "end effects" of a finite-interval smoother.

`FLSmovies.m` creates two short video clips of the smoothing improvement ratio $P[s]/P[f]$ as a function of QR/H^2 and FR/H^2 as the lag time parameter Δt_{lag} H^2/R varies from frame to frame from 10^{-6} to about 0.06. (It runs a long time.)

`RTSvsKF.m` is a MATLAB script for demonstrating the solutions to an estimation problem using

1. Kalman filtering, which uses only data up to the time of the estimate, and
2. Rauch–Tung–Striebel smoothing, which uses all the data.

A MATLAB implementation of a Rauch–Tung–Striebel smoother is included in the script, along with the corresponding Kalman filter implementation.

`BMFLS.m` implements the complete Biswas-Mahalanabis fixed-lag smoother (BMFLS) through the transient phase from the first measurement and through the transition to steady-state state augmentation and beyond.

`FPSperformance.m` computes the expected performance of fixed-point smoothing and plots the results.

A.8 MATLAB SOFTWARE FOR CHAPTER 6

`shootout.m` provides a demonstration of the relative fidelity of nine different ways to perform the covariance correction on Example 6.2.

To test how different solution methods perform as conditioning worsens, the observational update is performed for $10^{-9}\ \varepsilon^{2/3} \leq \delta \leq 10^{9}\ \varepsilon^{2/3}$ using nine different implementation methods:

1. The conventional Kalman filter, as published by R. E. Kalman;
2. Swerling inverse implementation, published by P. Swerling before the Kalman filter;

3. Joseph-stabilized implementation as given by P. D. Joseph;
4. Joseph-stabilized implementation as modified by G. J. Bierman;
5. Joseph-stabilized implementation as modified by T. W. DeVries;
6. The Potter algorithm (due to J. E. Potter);
7. The Carlson "triangular" algorithm (N. A. Carlson);
8. The Bierman *UD* algorithm (G. J. Bierman); and
9. The closed-form solution for this particular problem.

The first, second, and last methods are implemented within the m-file `shootout.m`. The others are implemented in m-files listed below.

The results are plotted as the RMS error in the computed value of P relative to the closed-form solution. In order that all results, including failed results, can be plotted, the value NaN (not a number) is interpreted as an underflow and set to zero, and the value Inf is interpreted as the result of a divide-by-zero and set to 10^4. This demonstration should show that, for this particular problem, the accuracies of the Carlson and Bierman implementations degrade more gracefully than the others as $\delta \rightarrow \varepsilon$. This might encourage the use of the Carlson and Bierman methods for applications with suspected roundoff problems, although it does not necessarily demonstrate the superiority of these methods for all applications:

`bierman.m` performs the Bierman *UD* implementation of the Kalman filter measurement update and

`carlson.m` performs the Carlson *fast triangular* implementation of the Kalman filter measurement update.

`joseph.m` performs the Joseph-stabilized implementation of the Kalman filter measurement update, as proposed by Bucy and Joseph [56];

`josephb.m` performs the Joseph-stabilized implementation of the Kalman filter measurement update, as modified by G. J. Bierman;

`josephdv.m` performs the Joseph-stabilized implementation of the Kalman filter measurement update, as modified by T. W. DeVries;

`potter.m` performs the Potter *square-root* implementation of the Kalman filter measurement update; and

`utchol.m` performs upper triangular Cholesky factorization for initializing the Carlson fast triangular implementation of the Kalman filter measurement update.

A.9 MATLAB SOFTWARE FOR CHAPTER 7

`ExamIEKF.m` performs Monte Carlo analysis of iterated extended Kalman filtering (EKF) on a nonlinear sensor problem.

`exam53.m` implements a discrete linearized Kalman filter.

`UTsimplex` implements the lowest-complexity unscented transform (UT) using "simplex" sampling. `UTsimplexDemo.m` is a MATLAB demonstration of its application.

`UTscaled.m` implements the scaled UT. `UTscaledDemo.m` demonstrates its use.

`UKFtempUD.m` implements the nonlinear Kalman filter temporal update using the UT.

A.10 MATLAB SOFTWARE FOR CHAPTER 8

`KFvsSKF.m` includes MATLAB implementations of the Schmidt–Kalman filter and Kalman filter for a common problem, implements both, and plots the results for comparison.

`thornton.m` implements the Thornton temporal update compatible with the Bierman observational update using modified Cholesky factors of P.

`Schmidt.m` performs the schmidt temporal update cumpatible with the Bierman observational update using modified Cholesky factors of P.

`InnovAnalysis.m` demonstrates innovations analysis on a linear stochastic process with nine different Kalman filters: one with the same model parameters as the simulated process and eight with the four model parameters Φ, H, Q, and R scaled up and down by a factor of 2.

A.11 MATLAB SOFTWARE FOR CHAPTER 9

`SchulerDemo.m` simulates and plots Schuler oscillations due to initial inertial navigation system (INS) errors.

`FNNmat.m` computes the dynamic coefficient matrix F_{NN} for INS navigation errors.

`Example9pt1.m` generates the plot used in Example 9.1.

`FSSmat.m` computes the dynamic coefficient matrix for inertial sensor errors.

`Example9pt3.m` simulates the nine-state INS navigation error dynamics.

`FSSQSmat.m` computes the dynamic coefficient submatrix F_{SS} and the covariance matrix QS for the INS sensor error model.

`FNSmat.m` computes the dynamic coefficient submatrix F_{NS} linking sensor errors to navigation errors.

`FreeInertial.m` calculates the propagation of INS navigation error covariances over a 10-h period, starting with zero navigation errors and the sensor error parameters shown in Table 9.2.

`Fig8TrackSim` generates values of vehicle dynamic state as it traverses a 1.5-km figure-8 test track at 90 kph.

`Fig8TrackSim2` computes additional values of vehicle dynamic state variables as it traverses a 1.5-km figure-8 test track at 90 kph.

`Example0901.m` demonstrates the effects of GNSS satellite motion on navigation performance.

`Example0906.m` demonstrates the influence of satellite geometry on GNSS navigation performance.

`YUMAdata.m` generates the actual GPS ephemeris information for Wednesday, March 8, 2006, at 10:48 A.M.

`HSatSim.m` uses the YUMA data to generate pseudorange measurement sensitivity matrices for GNSS receivers at a given latitude, longitude, and time.

`Example0906.m` requests four directions to GNSS satellites, then computes and plots navigation performance in terms of RMS uncertainties in position (three components) and clock errors (two parameters) versus time for 1 min.

`GNSSshootoutNCE.m` compares side-by-side the simulated performance of three different host vehicle dynamic models for GNSS navigation on a figure-8 racetrack and plots the results.

`AltStab.m` compares inertial navigation altitude estimates with and without auxiliary sensor damping.

`CDModelParams.m` computes dynamic model parameters Φ_θ and Q_θ for a critically damped system driven by white noise, given the system steady-state variances.

`HINS67` computes the 67-state measurement sensitivity matrix for an INS navigation sensor with 15 measurement variables for three components each of position, velocity, acceleration, attitude, and attitude rate.

`Fig8FreeInert.m` simulates navigation performance on a figure-8 track for a free-inertial INS without aiding (except for vertical channel stabilization).

`Fig8HVINS.m` calculates navigation performance on a figure-8 track for an INS without sensor aiding (except for vertical channel stabilization), but using a host vehicle dynamic model to statistically constrain navigation errors. This host vehicle dynamic model is "tuned" to the statistical dynamic conditions on the figure-8 test track. The same host vehicle model is used in the following performance simulators as well.

`Fig8GNSSonly.m` calculates expected performance using only GNSS navigation on a figure-8 track.

`Fig8GNSSINS.m` calculates and plots RMS performance statistics of a tightly coupled strapdown INS/GNSS integrated system which estimates and compensates for INS navigation errors and sensor errors.

A.12 OTHER SOURCES OF SOFTWARE

A.12.1 MATLAB Controls Toolbox

Available from The Mathworks, Controls Toolbox includes MATLAB routines for numerical solution of the algebraic Riccati equation for the Kalman filtering

problem for linear time-invariant systems. These essentially provide the steady-state Kalman gain (Wiener gain).

A.12.2 Software for Kalman Filter Implementation

There are several sources of good up-to-date software specifically designed to address the numerical stability issues in Kalman filtering. Scientific software libraries and workstation environments for the design of control and signal processing systems typically use the more robust implementation methods available. In addition, as a noncommercial source of algorithms for Kalman filtering, the documentation and source codes of the collected algorithms from the Transactions on Mathematical Software (TOMS) of the Association for Computing Machinery are available at moderate cost on electronic media. The TOMS collection contains several routines designed to address the numerical stability issues related to Kalman filter implementation, and these are often revised to correct deficiencies discovered by users.

Many sources for the unscented Kalman filter (UKF) can be found by searching the World Wide Web.

A.12.3 Utilities for Monte Carlo Simulation

The TOMS collection also contains several routines designed for pseudorandom number generation with good statistical properties. In addition, most reputable libraries contain good pseudorandom number generators, and many compilers include them as built-in functions. There are also several books (e.g., [221] or [128]) that come with the appropriate code on machine-readable media.

A.12.4 GNSS and Inertial Navigation Software

There are several sources of software for the design and analysis of inertial and GNSS navigation systems. These include:

1. Inertial Navigation System (INS) Toolbox for MATLAB
2. Navigation System Integration and Kalman Filter Toolbox for MATLAB
3. Satellite Navigation (SatNav) ToolBox for MATLAB

produced by GPSoft (www.gpsoftnav.com) and marketed through

Navtech Seminars & GPS Supply
www.navtechgps.com
Tel: 1-703-256-8900
Fax: 1-703-256-8988

Other commercial sources can be located by searching the World Wide Web.

APPENDIX B

A MATRIX REFRESHER

This overview of the notation and properties of matrices as data structures and algebras is for readers familiar with the general subject of linear algebra but whose recall may be a little rusty. A more thorough treatment can be found in most college-level textbooks on linear algebra and matrix theory.

B.1 MATRIX FORMS

B.1.1 Notation for Real Matrices

B.1.1.1 Scalars For the purposes of this book, *scalars* are real numbers, although in computer implementations they must be approximated by floating-point numbers, which are but a finite subset of the rational numbers. We will use parentheses to denote open intervals (intervals not including the designated endpoints) on the real line, so that $(-\infty, +\infty)$ denotes the set of all real numbers. We will use square brackets to denote closed ends (ends including the designated endpoint) of intervals, so that $[0,+\infty)$ denotes the nonnegative real numbers.

B.1.1.2 Real Matrices For positive integers m and n, an m-by-n real *matrix* A is a two-dimensional rectangular array of scalars, designated by the subscript notation a_{ij}

Kalman Filtering: Theory and Practice using MATLAB®, Third Edition. By Mohinder S. Grewal and Angus P. Andrews

and usually displayed in the following format:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{bmatrix}.$$

The scalars a_{ij} are called the *elements* of A. We will use uppercase letters to denote matrices and the corresponding lowercase letters to denote scalar elements of the associated matrices.

B.1.1.3 Indices and Subscripts The first subscript (i) on the element a_{ij} refers to the *row* in which the element occurs, and the second subscript (j) refers to the *column* in which a_{ij} occurs in this format. The integers i and j in this notation are also called *indices* of the elements. The first index is called the *row index*, and the second index is called the *column index* of the element. The term (ij)th *position* in the matrix A refers to the position of a_{ij}, and a_{ij} is called the (ij)th *element* of A.

If juxtaposition of subscripts leads to confusion, they may be separated by commas. The element in the 11th row and first column of the matrix A would then be denoted by $a_{11,1}$, not a_{111}.

B.1.1.4 Dimensions The positive integers m and n are called the *dimensions* of A: m is called the *row dimension* of A and n is called the *column dimension* of A. The dimensions of A may also be represented as $m \times n$, which is to be read "m by n." The symbol "$\times$" in this notation does not indicate multiplication. (The number of elements in the matrix A equals the product mn, however, and this is important for determining memory requirements for data structures to hold A.)

B.1.2 Special Matrix Forms

B.1.2.1 Square Matrices and Diagonal Matrices A matrix is called *square* if it has the same row and column dimensions. The *main diagonal* of a square matrix A is the set of elements a_{ij} for which $i = j$. The other elements are called *off-diagonal*. If all the off-diagonal elements of a square matrix A are zero, A is called a *diagonal* matrix. This and other special forms of square matrices are illustrated in Figure B.1.[1]

B.1.2.2 Sparse and Dense Matrices A matrix with a "significant fraction" (typically, half or more) of zero elements is called *sparse*. Matrices that are decidedly not

[1]The matrix forms in the third row of Figure B.1 belong to both forms in the column above. That is, diagonal matrices are both upper triangular and lower triangular, identity matrices are both unit upper triangular and unit lower triangular, and square zero matrices are both strictly upper triangular and strictly lower triangular.

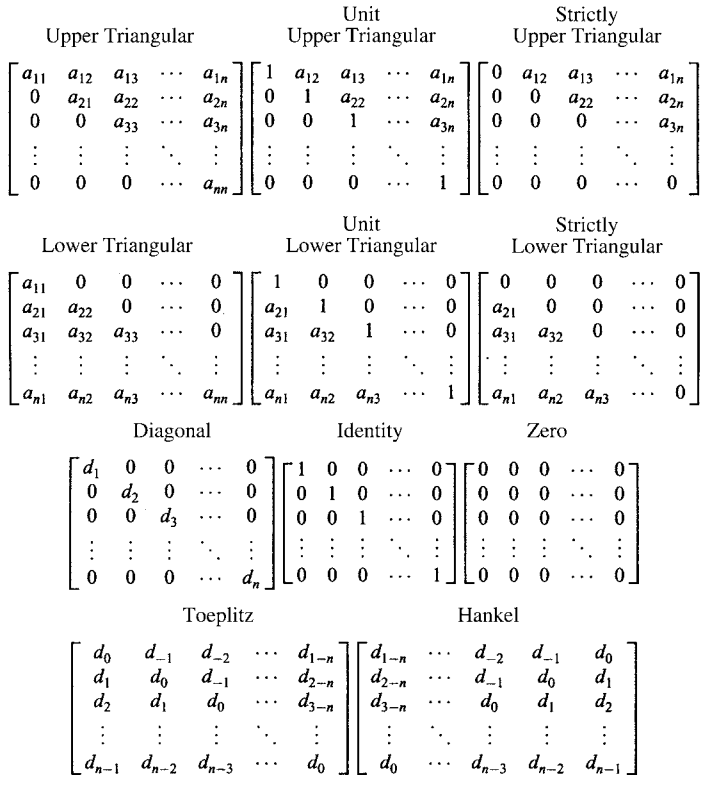


Figure B.1 Special forms of square matrices.

sparse are called *dense*, although both sparsity and density are matters of degree. Except for the Toeplitz and Hankel matrices,[2] the forms shown in Figure B.1 are sparse, although sparse matrices do not have to be square. Sparsity is an important characteristic for implementation of matrix methods, because it can be exploited to reduce computer memory and computational requirements.

B.1.2.3 Zero Matrices The ultimate sparse matrix is a matrix in which *all* elements are 0 (zero). It is called a *zero matrix*, and it is represented by the symbol "0" (zero). The equation $A = 0$ indicates that A is a zero matrix. Whenever it is necessary to specify the dimensions of a zero matrix, they may be indicated by subscripting: $0_{m\times n}$ will indicate an $m \times n$ zero matrix. If the matrix is square, only one subscript will be used: 0_n will mean an $n \times n$ zero matrix.

[2]Although a Toeplitz matrix is fully dense, it can be represented by the $2n-1$ distinct values of its elements.

B.1.2.4 Identity Matrices The identity matrix will be represented by the symbol I. If it is necessary to denote the dimension of I explicitly, it will be indicated by subscripting the symbol: I_n denotes the $n \times n$ identity matrix.

B.1.3 Vectors

B.1.3.1 Vectors and Matrices A *vector* is essentially a matrix[3] in which one of the dimensions is 1. If the column dimension is 1, it is called a *column vector*. If the row dimension is 1, it is called called a *row vector*.[4] Once it is understood which of the dimensions is 1, the index for that dimension can be dropped from the notation. The other dimension may be prefixed to *-vector* to shorten the notation: the term *n-vector* refers to a matrix for which one dimension is 1 and the other is n.

B.1.3.2 Representational Differences Although vectors may share many mathematical properties with matrices, they are used in this book to represent quite different concepts. To a considerable degree, vectors are used to represent physically measurable properties of dynamic systems, and matrices are used to represent transformations of those properties performed by sensors or by the passage of time. In order to make the distinction between vectors and matrices more apparent, we shall use lowercase letters to denote vectors and scalars and uppercase letters to denote matrices. However, as data structures, vectors are not fundamentally different from matrices or other arrays.

B.1.3.3 Row Vector Notation Commas can be used for separating the elements of a row vector:

$$x = [x_1, x_2, x_3, \ldots, x_n],$$

where the notation "x_i" refers to the element in the ith column of x. (We will return to a related issue—the compositional efficiency of row vector notation—after the matrix transpose is defined.)

B.1.3.4 Column Vector Default Whenever a vector x is not defined to be a row vector, it is implied that it is a column vector.

B.1.4 Conformable Matrix Dimensions

B.1.4.1 Syntax of Mathematical Expressions *Syntax* is a set of rules governing the formation of patterns of symbols in a language. In mathematics, these symbols may stand for data structures such as scalars and matrices, operators such as addition and multiplication, or delimiters such as parentheses and brackets. Patterns of these

[3]Defining a vector as a special case of a matrix is not the customary approach, but it obviates the need for separate definitions of "inner" and "outer" products.

[4]And if *both* dimensions are 1, it is called a *scalar*, not a *1-vector*.

symbols satisfying the rules of syntax are called *expressions*. Within the constraints imposed by syntax, an expression of a particular type (e.g., a scalar or a matrix expression) can be substituted for a symbol of that type, and vice versa.

B.1.4.2 Syntax for Matrix Dimensions For matrix expressions, there are additional rules of syntax related to the dimensions of the matrices. For example, whenever we write a matrix equation as $A = B$, we assume that the matrices (or matrix expressions) represented by the symbols A and B have the same dimensions.

B.1.4.3 Implied Conformability of Matrix Dimensions Additional rules of syntax for matrix operations will be introduced with the operators. Matrices whose dimensions conform to these rules for a position in a particular expression are said to be *conformable* for that position in that expression. Whenever a symbol for a matrix appears in a matrix expression, it is implied that the matrix represented by that symbol is conformable for its position in that expression.

B.2 MATRIX OPERATIONS

B.2.1 Transposition

B.2.1.1 Transpose of a Matrix All matrices are conformable for transposition. The *transpose* of A is the matrix A^{T} (with the superscript "T" denoting the transpose operation), obtained from A by interchanging rows and columns:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} a_{11} & a_{21} & a_{31} & \cdots & a_{m1} \\ a_{12} & a_{22} & a_{32} & \cdots & a_{m2} \\ a_{13} & a_{23} & a_{33} & \cdots & a_{m3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & a_{3n} & \cdots & a_{mn} \end{bmatrix}.$$

The transpose of an $m \times n$ matrix is an $n \times m$ matrix.

B.2.1.2 Transpose of a Vector The transpose of a row vector is a column vector, and vice versa. It makes more efficient use of space on the page if we express a column vector v as the transpose of a row vector:

$$v = [v_1, v_2, v_3, \ldots, v_m]^{\mathrm{T}}.$$

B.2.1.3 Symmetric Matrices A matrix A is called *symmetric* if $A^{\mathrm{T}} = A$ and *skew symmetric* (or *antisymmetric*) if $A^{\mathrm{T}} = -A$. Only square matrices can be symmetric or skew symmetric. Therefore, whenever a matrix is said to be symmetric or skew symmetric, it is implied that it is a square matrix.

B.2.2 Extracting Elements of Matrix Expressions

B.2.2.1 Subscripted Expressions Subscripts represent an operation on a matrix that extracts the designated matrix element. Subscripts may also be applied to matrix expressions. The element in the (ij)th position of a matrix expression can be indicated by subscripting the expression, as in

$$\{A^{\mathrm{T}}\}_{ij} = a_{ji}.$$

Here, we have used braces $\{\}$ to indicate the scope of the expression to which the subscripting applies. This is a handy device for defining matrix operations.

B.2.3 Multiplication by Scalars

All matrices are conformable for multiplication by scalars, either on the left or on the right. Multiplication is indicated by juxtaposition of symbols or by infix notation with the multiplication symbol ($\times$). Multiplication of a matrix A by a scalar s is equivalent to multiplying every element of A by s:

$$\{As\}_{ij} = \{sA\}_{ij} = sa_{ij}.$$

B.2.4 Addition and Multiplication of Conformable Matrices

B.2.4.1 Addition of Conformable Matrices Is Associative and Commutative Matrices are conformable for addition if and only if they share the same dimensions. Whenever matrices appear as sums in an expression, it is implied that they are conformable. If A and B are conformable matrices, then addition is defined by adding corresponding elements:

$$\{A + B\}_{ij} = a_{ij} + b_{ij}.$$

Addition of matrices is *commutative* and *associative*. That is, $A + B = B + A$ and $A + (B + C) = (A + B) + C$.

B.2.4.2 Additive Inverse of a Matrix The product of a matrix A by the scalar -1 yields its *additive inverse* $-A$:

$$(-1)A = -A, \quad A + (-A) = A - A = 0.$$

Here, we have followed the not uncommon practice of using the symbol "$-$" both as a unary (additive inverse) and a binary (subtraction) operator. *Subtraction* of a matrix A from a matrix B is equivalent to adding the additive inverse of A to B:

$$B - A = B + (-A).$$

B.2.4.3 Multiplication of Conformable Matrices Is Associative But Not Commutative Multiplication of an $m \times n$ matrix A by a matrix B on the right-hand side of A, as in the matrix product AB, is defined only if *the row dimension of B equals the column dimension of A*. That is, we can multiply an $m \times n$ matrix A by a $p \times q$ matrix B in this order only if $n = p$. In that case, the matrices A and B are said to be conformable for multiplication in that order, and the matrix product is defined element by element by

$$\{AB\}_{ij} \stackrel{\text{def}}{=} \sum_{k=1}^{n} a_{ik} b_{kj},$$

the result of which is an $m \times q$ matrix. Whenever matrices appear as a product in an expression, it is implied that they are conformable for multiplication.

B.2.4.4 Inner and Outer Products Special names are given to products of vectors that are otherwise adequately defined as matrix products. The *inner product* or *dot product* of conformable column vectors x and y is the matrix product $x^{\mathrm{T}}y = y^{\mathrm{T}}x$. (For row vectors, the format is $xy^{\mathrm{T}} = yx^{\mathrm{T}}$.) The *outer products* have the transpose on the other vector, and the vectors need not have the same dimensions. If the vectors are treated as matrices, these products can be used without special treatment.

B.2.4.5 Products with Identity Matrices Multiplication of any m x n matrix A by a conformable *identity matrix* yields the original matrix A as the product:

$$AI_n = A, \quad I_m A = A.$$

B.2.5 Powers of Square Matrices

Square matrices are conformable with multiplication by themselves, and the resulting matrix products are again conformable for multiplication. Consequently, one can define the pth *power* of a square matrix A as

$$A^p = \underbrace{A \times A \times A \times \cdots \times A}_{p \text{ elements}}.$$

B.2.6 Matrix Inverses

B.2.6.1 Inverses of Nonsingular Square Matrices If A and B are square matrices of the same dimension and are such that their product

$$AB = I,$$

then B is the *matrix inverse* of A and A is the matrix inverse of B. (It turns out that $BA = AB = I$ in this case.) The inverse of a matrix A is unique, if it exists, and is

denoted by A^{-1}. Not all matrices have inverses. *Matrix inversion* is the process of finding a matrix inverse if it exists. If the inverse of a matrix A does not exist, A is called *singular*. Otherwise, it is called *nonsingular*.

B.2.6.2 Generalized Inverses Even nonsquare or singular matrices can have *generalized inverses*. The *Moore–Penrose*[5] *generalized inverse* of an $m \times n$ matrix A is the $n \times m$ matrix $A^{\dagger}$ such that

$$AA^{\dagger}A = A, \quad A^{\dagger}AA^{\dagger} = A^{\dagger},$$
$$(AA^{\dagger})^{\mathrm{T}} = AA^{\dagger}, \quad (A^{\dagger}A)^{\mathrm{T}} = A^{\dagger}A.$$

The MATLAB function `pinv(M)` computes the Moore–Penrose generalized inverse of the matrix M using the singular value decomposition of M. This result is sometimes called the *pseudoinverse* of M, because it is based on the *pseudorank* of M, which is the effective rank within machine precision tolerance.

B.2.7 Orthogonality

B.2.7.1 Orthogonal Vectors For vectors, *orthogonality* is a pairwise property. Vectors x and y are called *orthogonal* or *normal* if their inner product is 0. If the inner product of a vector x with itself is 1, x is called a *unit vector*. Orthogonal unit vectors are called *orthonormal*.[6]

B.2.7.2 Orthogonal Matrices A matrix A is called orthogonal if $A^{\mathrm{T}} = A^{-1}$. These matrices have several useful properties:

- Orthogonality of a matrix A implies that the row vectors of A are jointly orthonormal and the column vectors of A are also jointly orthonormal.
- The dot products of vectors are invariant under multiplication by a conformable orthogonal matrix. That is, if A is orthogonal, then $x^{\mathrm{T}}y = (Ax)^{\mathrm{T}}(Ay)$ for all conformable x and y.
- Products and inverses of orthogonal matrices are orthogonal.

As a rule, multiplications by orthogonal matrices tend to be numerically well conditioned—compared to general matrix multiplications. (The inversion of orthogonal matrices is obviously extremely well conditioned.)

[5]Named for Eliakim Hastings Moore (1862–1932) and Sir Roger Penrose (b. 1931).

[6]The term *normal* has many meanings in mathematics. Its use here is synonymous with *of unit length*. It is also used to mean "orthogonal."

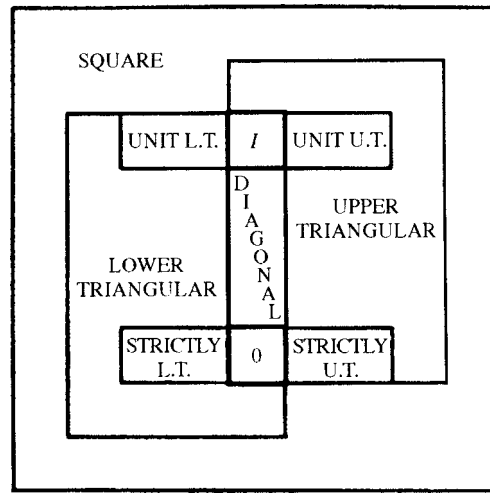


Figure B.2 Subalgebras of square matrices.

B.2.8 Square Matrix Subalgebras

Certain subclasses of $n \times n$ (square) matrices have the property that their products belong to the same class. Orthogonal matrices, for example, have the property that their products are also orthogonal matrices. These are said to form a multiplicative *subalgebra* of $n \times n$ matrices. Subalgebras have the property that their set intersections are also subalgebras. Upper triangular matrices and lower triangular matrices are two subalgebras of the square matrices that are used in implementing the Kalman filter. Their intersection is the set of diagonal matrices—another subalgebra. A lattice (partially ordered by set inclusion) of such multiplicative subalgebras is diagrammed in Figure B.2.

B.3 BLOCK MATRIX FORMULAS

B.3.1 Submatrices, Partitioned Matrices, and Blocks

For any $m \times n$ matrix A and any subset $S_{\text{rows}} \subseteq \{1, 2, 3, \ldots, m\}$ of the row indices and subset $S_{\text{cols}} \subseteq \{1, 2, 3, \ldots, n\}$ of the column indices, the subset of elements

$$A' = \{a_{ij} | i \in S_{\text{rows}}, j \in S_{\text{cols}}\}$$

is called a *submatrix* of A.

A *partitioning* of an integer n is an exhaustive collection of contiguous subsets S_k of the form

$$\overbrace{1, 2, 3, \ldots, \ell_1}^{S_1}, \overbrace{(\ell_1 + 1), \ldots, \ell_2}^{S_2}, \ldots, \overbrace{(\ell_{p-1} + 1), \ldots, n}^{S_p}.$$

The collection of submatrices formed by partitionings of the row and column dimensions of a matrix is called a *partitioning of the matrix*, and the matrix is said to be *partitioned by that partitioning*. Each submatrix of a partitioned matrix A is called a *partitioned submatrix*, *partition*, *submatrix block*, *subblock*, or *block* of A. Each block of a partitioned matrix A can be represented by a conformable matrix expression, and A can be displayed as a *block matrix*:

$$A = \begin{bmatrix} B & C & D & \cdots & F \\ G & H & J & \cdots & L \\ M & N & P & \cdots & R \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ V & W & X & \cdots & Z \end{bmatrix},$$

where B, C, D, ... stand for matrix expressions. Whenever a matrix is displayed as a block matrix, it is implied that all block submatrices in the same row have the same row dimension and that all block submatrices in the same column have the same column dimension.

A block matrix of the form

$$\begin{bmatrix} A & 0 & 0 & \cdots & 0 \\ 0 & B & 0 & \cdots & 0 \\ 0 & 0 & C & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & M \end{bmatrix},$$

in which the off-diagonal block submatrices are zero matrices, is called a *block diagonal matrix*, and a block matrix in which the block submatrices on one side of the diagonal are zero matrices is called a *block triangular matrix*.

B.3.1.1 Columns and Rows as Blocks There are two special partitionings of matrices in which the block submatrices are vectors. The column vectors of an $m \times n$ matrix A are the block submatrices of the partitioning of A for which all column dimensions are 1 and all row dimensions are m. The row vectors of A are the block submatrices of the partitioning for which all row dimensions are 1 and all column dimensions are n. All column vectors of an $m \times n$ matrix are m-vectors, and all row vectors are n-vectors.

B.3.2 Rank and Linear Dependence

A *linear combination* of a finite set of n-vectors $\{v_i\}$ is a summation of the sort $\sum_i a_i v_i$ for some set of scalars $\{a_i\}$. If some linear combination $\sum a_i v_i = 0$ and at least one coefficient $a_i \neq 0$, the set of vectors $\{v_i\}$ is called *linearly dependent*. Conversely, if the only linear combination for which $\sum a_i v_i = 0$ is the one for which all the $a_i = 0$, then the set of vectors $\{v_i\}$ is called *linearly independent*.

The *rank* of a $n \times m$ matrix A equals the size of the *largest* collection of its column vectors that is linearly independent. Note that any such linear combination can be expressed in the form Aa, where the nonzero elements of the column m-vector a are the associated scalars of the linear combination, and the number of nonzero components of a is the size of the collection of column vectors in the linear combination. The same value for the rank of a matrix is obtained if the test is applied to its row vectors, where any linear combination of row vectors can be expressed in the form $a^{\mathrm{T}}A$ for some column n-vector a.

An $n \times n$ matrix is nonsingular if and only if its rank equals its dimension n.

B.3.3 Conformable Block Operations

Block matrices with conformable partitionings may be transposed, added, subtracted, and multiplied in block format. For example,

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} A^{\mathrm{T}} & C^{\mathrm{T}} \\ B^{\mathrm{T}} & D^{\mathrm{T}} \end{bmatrix},$$

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} + \begin{bmatrix} E & F \\ G & H \end{bmatrix} = \begin{bmatrix} A+E & B+F \\ C+G & D+H \end{bmatrix},$$

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \times \begin{bmatrix} E & F \\ G & H \end{bmatrix} = \begin{bmatrix} AE+BG & AF+BH \\ CE+DG & CF+DH \end{bmatrix}.$$

B.3.4 Frobenius–Schur Inversion Formula

The inverse of a partitioned matrix with square diagonal blocks may be represented in block form as[7]

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}^{-1} = \begin{bmatrix} E & F \\ G & H \end{bmatrix},$$

[7]This formula has had many discoverers. Bodewig [42] cites nine such discoverers but gives credit to the German mathematicians Georg Ferdinand Frobenius (1849–1917) and Issai Schur (1875–1941) as the earliest discovers of record.

where

$$E = A^{-1} + A^{-1}BHCA^{-1},$$
$$F = -A^{-1}BH,$$
$$G = -HCA^{-1},$$
$$H = [D - CA^{-1}B]^{-1}.$$

This formula can be proved by multiplying the original matrix by its alleged inverse and verifying that the result is the identity matrix.

B.3.5 Inversion Formulas for Matrix Expressions

B.3.5.1 Sherman–Morrison Formula[8] A *rank 1* modification of a square matrix A is a sum of the form $A + bc^{\mathrm{T}}$, where b and c are conformable column vectors.[9] Its inverse is given by the formula

$$[A + bc^{\mathrm{T}}]^{-1} = A^{-1} - \frac{A^{-1}bc^{\mathrm{T}}A^{-1}}{1 + c^{\mathrm{T}}A^{-1}b}.$$

B.3.5.2 Sherman–Morrison–Woodbury Formula This is the generalization of the above formula for conformable matrices in place of vectors:

$$[A + BC^{\mathrm{T}}]^{-1} = A^{-1} - A^{-1}B[I + C^{\mathrm{T}}A^{-1}B]^{-1}C^{\mathrm{T}}A^{-1}.$$

B.3.5.3 Hemes Inversion Formula A further generalization of this formula (used in the derivation of the Kalman filter equations) includes an additional conformable square matrix factor in the modification[10]

$$[A + BC^{-1}D^{\mathrm{T}}]^{-1} = A^{-1} - A^{-1}B[C + D^{\mathrm{T}}A^{-1}B]^{-1}D^{\mathrm{T}}A^{-1}.$$

[8]The naming of this and the next formula follows the convention of Golub and Van Loan [99], which is at odds with the references of Henderson and Searle [110].

[9]A finite set of vectors $\{x_1, x_2, x_3, \ldots, x_n\}$ is said to be *linearly independent* if there is no linear combination $\sum_k a_k x_k = 0$ with no coefficient $a_k = 0$. The rank of a set of vectors is defined as the size of the maximum subset of them that is linearly independent. The rank of the row vectors of a matrix is called its *row rank*, and the rank of its column vectors is called its *column rank*. For a square matrix, the row rank equals the column rank. An $n \times n$ square matrix is nonsingluar if and only if its (row *or* column) rank is n. Obviously, the rank of a single vector is 1.

[10]This is yet another formula with many discoverers. Fortmann [87] cites several of them. Bodewig [42, p. 218] credits H. Hemes for its discovery, although Henderson and Searle [110] cite an earlier reference.

B.4 FUNCTIONS OF SQUARE MATRICES

B.4.1 Determinants and Characteristic Values

B.4.1.1 Elementary Permutation Matrices An *elementary permutation matrix* is formed by interchanging rows or columns of an identity matrix I_n:

$$P_{[ij]} = \begin{array}{c} \\ i \\ \\ j \\ \\ \end{array} \begin{array}{c} \begin{array}{ccccccc} & & i & & j & & \end{array} \\ \begin{pmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & 0 & \cdots & 1 & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & 1 & \cdots & 0 & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{pmatrix} \end{array}.$$

Multiplication of a vector x by $\mathcal{P}_{[ij]}$ permutes the ith and jth elements of x. Note that $P_{[ij]}$ is an *orthogonal* matrix and $P_{[ij]} = I_n$ is the identity matrix.

B.4.1.2 Determinants of Elementary Permutation Matrices The *determinant* of an elementary permutation matrix $P_{[ij]}$ is defined to be -1 unless $i = j$ (i.e., $P_{[ij]} = I_n$):

$$\det(P_{[ij]}) \stackrel{\text{def}}{=} \begin{cases} -1, & i \neq j, \\ +1, & i = j. \end{cases}$$

B.4.1.3 Permutation Matrices A *permutation matrix* is any product of elementary permutation matrices. These are also orthogonal matrices. Let $\mathcal{P}_n$ denote the set of all distinct $n \times n$ permutation matrices. There are $n! = 1 \times 2 \times 3 \times \cdots \times n$ of them, corresponding to the $n!$ permutations of n indices.

B.4.1.4 Determinants of Permutation Matrices The determinant of a permutation matrix can be defined by the rule that the determinant of a product of matrices is the product of the determinants:

$$\det(AB) = \det(A)\det(B).$$

Therefore, the determinant of a permutation matrix will be either $+1$ or -1. A permutation matrix is called *even* if its determinant is $+1$ and *odd* if its determinant equals -1.

B.4.1.5 Determinants of Square Matrices The determinant of any $n \times n$ matrix A can be defined as follows:

$$\det(A) \stackrel{\text{def}}{=} \sum_{P \in \mathcal{P}_n} \det(P) \prod_{i=1}^{n} \{AP\}_{ii}.$$

This formula has $\mathcal{O}(n \times n!)$ computational complexity (for a sum over $n!$ products of n elements each).

B.4.1.6 Characteristic Values of Square Matrices For a free variable λ, the polynomial

$$\begin{aligned} P_A(\lambda) &\stackrel{\text{def}}{=} \det[A - \lambda I] \\ &= \sum_{i=0}^{n} a_i \lambda^i \end{aligned}$$

is called the *characteristic polynomial* of A. The roots of $p_A(\lambda)$ are called the *characteristic values* (or *eigenvalues*) of A. The determinant of A equals the product of its characteristic values, with each characteristic value occurring as many times in the product as the multiplicity of the associated root of the characteristic polynomial.

B.4.1.7 Definiteness of Symmetric Matrices If A is symmetric, all its characteristic values are real numbers, which implies that they can be ordered. They are usually expressed in descending order:

$$\lambda_1(A) \geq \lambda_2(A) \geq \lambda_3(A) \geq \cdots \geq \lambda_n(A).$$

A real square symmetric matrix A is called

positive definite	if $\lambda_n(A) > 0$,
nonnegative definite	if $\lambda_n(A) \geq 0$,
indefinite	if $\lambda_1(A) < 0$ and $\lambda_n(A) > 0$,
nonpositive definite	if $\lambda_1(A) \leq 0$, and
negative definite	if $\lambda_1(A) < 0$.

Nonnegative-definite matrices are also called *positive semidefinite*, and nonpositive-definite matrices are also called *negative semidefinite*.

B.4.1.8 Characteristic Vectors For each real characteristic value $\lambda_i(A)$ of a real symmetric A, there is a corresponding *characteristic vector* (or *eigenvector*) $e_i(A)$

such that $e_i(A) \neq 0$ and $Ae_i(A) = \lambda_i(A)e_i(A)$. The characteristic vectors corresponding to distinct characteristic values are mutually orthogonal.

B.4.2 Matrix Trace

The *trace* of a square matrix is the sum of its diagonal elements. It also equals the sum of the characteristic values and has the property that the trace of the product of conformable matrices is independent of the order of multiplication—a very useful attribute:

$$\text{trace}(AB) = \sum_i \{AB\}_{ii} \tag{B.1}$$

$$= \sum_i \sum_j A_{ij}B_{ji} \tag{B.2}$$

$$= \sum_j \sum_i B_{ji}A_{ij} \tag{B.3}$$

$$= \text{trace}(BA). \tag{B.4}$$

Note that product AB is conformable for the trace function only if it is a square matrix, which requires that A and B^T have the same dimensions. If they are $m \times n$ (or $n \times m$), then the computation of the trace of their product requires mn multiplications, whereas the product itself would require m^2n (or mn^2) multiplications.

B.4.3 Algebraic Functions of Matrices

An algebraic function may be defined by an expression in which the independent variable (a matrix) is a free variable, such as the truncated power series

$$f(A) = \sum_{k=-n}^{n} B_k A^k,$$

where the negative power $A^{-p} = \{A^{-1}\}^{\text{p}} = \{A^p\}^{-1}$. In this representation, the matrix A is the independent (free) variable and the other matrix parameters (B_k) are assumed to be known and fixed.

B.4.4 Analytic Functions of Matrices

An analytic function is defined in terms of a convergent power series. It is necessary that the power series converge to a limit, and the matrix norms defined in Section B.5 must be used to define and prove convergence of a power series. This level of rigor is beyond the scope of this book, but we do need to use one particular analytic function—the exponential function.

B.4.4.1 Exponential Function The power series

$$e^A = \sum_{k=0}^{\infty} \frac{1}{1 \cdot 2 \cdot 3 \cdots k} A^k$$

does converge for all square matrices A, although convergence is not fast enough to make this a reasonable general-purpose formula for approximating the exponential of A. This definition does suffice to prove some elementary properties of the exponential function for matrices, such as:

- $e^{0_n} = I_n$ for 0_n the $n \times n$ zero matrix.
- $e^{I_n} = eI_n$ for I_n the $n \times n$ identity matrix.
- $e^{A^T} = \{e^A\}^T$.
- $(d/dt)e^{At} = Ae^{At} = e^{At}A$.
- The exponential of a skew-symmetric matrix is an orthogonal matrix.
- The characteristic vectors of A are also the characteristic vectors of e^A.
- If λ is a characteristic value of A, then e^λ is a characteristic value of e^A.

B.4.4.2 Computing Matrix Exponentials In a 1978 journal article titled "Nineteen Dubious Ways to Compute the Exponential of a Matrix" [197], Moler and Van Loan reported their evaluations of methods for computing matrix exponentials. Many of the methods tested had serious shortcomings, and no method was declared universally superior. The one presented here was recommended as being more reliable than most. It combines several ideas due to Ward [280], including setting the algorithm parameters to meet a prespecified error bound. It combines Padé approximation with a technique called *scaling and squaring* to maintain approximation errors within prespecified bounds.

Padé Approximation of the Matrix Exponential Padé approximations of functions by rational functions (ratios of polynomials) date from a 1892 publication [207] by Henri Padé.[11] They have been used in deriving solutions of differential equations, including Riccati equations[12] [178]. They can also be applied to functions of matrices, including the matrix exponential. In the matrix case, the power series is approximated as a "matrix fraction" of the form $\mathcal{D}^{-1}\mathcal{N}$, with the numerator matrix ($\mathcal{N}$) and denominator matrix ($\mathcal{D}$) represented as polynomials with matrix arguments. The "order" of the Padé approximation is two-dimensional. It depends on the orders of the polynomials in the numerator and denominator of the rational function. The Taylor series is the special case in which the order of the denominator polynomial of the Padé approximation is zero. Like the Taylor series approximation, the Padé

[11] Pronounced *pah-DAY*.

[12] The order of the numerator and denominator of the matrix fraction are reversed here from the order used in linearizing the Riccati equation in Chapter 4.

approximation tends to work best for small values of its argument. For matrix arguments, it will be some matrix norm[13] of the argument that will be required to be small.

The exponential function with argument z has the power series expansion

$$e^z = \sum_{k=0}^{\infty} \frac{1}{k!} z^k.$$

The polynomials $\mathcal{N}_p(z)$ and $\mathcal{D}_q(z)$ such that

$$\mathcal{N}_p(z) = \sum_{k=0}^{p} a_k z^k,$$

$$\mathcal{D}_q(z) = \sum_{k=0}^{q} b_k z^k,$$

$$e^z \mathcal{D}_q(z) - \mathcal{N}_p(z) = \sum_{k=p+q+1}^{\infty} c_k z^k$$

are the numerator and denominator polynomials, respectively, of the Padé approximation of e^z. The key feature of the last equation is that there are no terms of order $\leq p + q$ on the right-hand side. This constraint is sufficient to determine the coefficients a_k and b_k of the polynomial approximants except for a common constant factor. The solution (within a common constant factor) will be [178]

$$a_k = \frac{p!(p+q-k)!}{k!(p-k)!}, \quad b_k = \frac{(-1)^k q!(p+q-k)!}{k!(q-k)!}.$$

Application to Matrix Exponential The above formulas may be applied to polynomials with scalar coefficients and square matrix arguments. For any $n \times n$ matrix X,

$$f_{pq}(X) = \left(q! \sum_{i=0}^{q} \frac{(p+q-i)!}{i!(q-i)!} (-X)^i \right)^{-1} \left(p! \sum_{i=0}^{p} \frac{(p+q-i)!}{i!(p-i)!} X^i \right)$$
$$\approx e^X$$

is the Padé approximation of e^X of order (p, q).

[13] Defined in Section B.5.

Bounding Relative Approximation Error The bound given here is from Moler and Van Loan [197]. It uses the ∞-norm of a matrix, which can be computed[14] as

$$\|X\|_\infty = \max_{1\le i\le n}\left(\sum_{j=1}^{n} |x_{ij}|\right)$$

for any $n \times n \times$ matrix X with elements x_{ij}. The *relative* approximation error is defined as the ratio of the matrix ∞-norm of the approximation error to the matrix ∞-norm of the right answer. The relative Padé approximation error is derived as an analytical function of X in Moler and Van Loan [197]. It is shown in Golub and Van Loan [99] that it satisfies the inequality bound

$$\frac{\|f_{pq}(X) - e^X\|_\infty}{\|e^X\|_\infty} \le \varepsilon(p,q,X)e^{\varepsilon(p,q,X)},$$

$$\varepsilon(p,q,X) = \frac{p!q!2^{3-p-q}}{(p+q)!(p+q+1)!}\|X\|_\infty.$$

Note that this bound depends only on the *sum* $p + q$. In that case, the computational complexity of the Padé approximation for a given error tolerance is minimized when $p = q$, that is, if the numerator and denominator polynomials have the same order.

The problem with the Padé approximation is that the error bound grows exponentially with the norm $\|X\|_\infty$. Ward [280] combined scaling (to reduce $\|X\|_\infty$ and the Padé approximation error) with squaring (to rescale the answer) to obtain an approximation with a predetermined error bound. In essence, one chooses the polynomial order to achieve the given bound.

Scaling and Squaring Note that, for any nonnegative integer N,

$$\begin{aligned} e^X &= (e^{2^{-N}X})^{2^N} \\ &= \underbrace{\{[(\cdots e^{2^{-N}X}\cdots)^2]^2\}^2}_{N \text{ squarings}}. \end{aligned}$$

Consequently, X can be "downscaled" by 2^{-N} to obtain a good Padé approximation of $e^{2^{-N}X}$, then "upscaled" again (by N squarings) to obtain a good approximation to e^X.

MATLAB Implementations The built-in MATLAB function `expm(M)` is essentially the one recommended by Moler and Van Loan [197], as implemented by

[14]This formula is not the definition of the ∞-norm of a matrix, which is defined in Section B.5.2. However, it is a consequence of the definition, and it can be used for computing it.

Golub and Van Loan [99], Algorithm 11.3.1, p. 558]. It combines scaling and squaring with a Padé approximation for the exponential of the scaled matrix, and it is designed to achieve a specified tolerance of the approximation error. The MATLAB m-file `expm1.m` (Section A.4) is a script implementation of `expm`.

MATLAB also includes the functions `expm2` (Taylor series approximation) and `expm3` (alternative implementation using eigenvalue–eigenvector decompositions), which can be used to test the relative accuracies and speeds relative to `expm` of these alternative implementations of the matrix exponential function.

B.4.5 Similarity Transformations and Analytic Functions

For any $n \times n$ nonsingular matrix A, the transform $X \rightarrow A^{-1}XA$ is called a *similarity transformation* of the $n \times n$ matrix X. It is a useful transformation for analytic functions of matrices

$$f(X) = \sum_{k=0}^{\infty} a_k X^k,$$

because

$$\begin{aligned} f(A^{-1}XA) &= \sum_{k=0}^{\infty} a_k (A^{-1}XA)^k \\ &= A^{-1}\left(\sum_{k=0}^{\infty} a_k X^k\right)A \\ &= A^{-1}f(X)A. \end{aligned}$$

If the characteristic values of X are distinct, then the similarity transform performed with the characteristic vectors of X as the column vectors of A will diagonalize X with its characteristic values along the main diagonal:

$$\begin{aligned} A^{-1}XA &= \operatorname*{diag}_{\ell}\{\lambda_\ell\}, \\ f(A^{-1}XA) &= \operatorname*{diag}_{\ell}\{F(\lambda_\ell)\}, \\ f(X) &= A \operatorname*{diag}_{\ell}\{F(\lambda_\ell)\}A^{-1}. \end{aligned}$$

(Although this is a useful analytical approach for demonstrating functional dependencies, it is not considered a robust numerical method.)

B.5 NORMS

B.5.1 Normed Linear Spaces

Vectors and matrices can be considered as elements of *linear spaces* in that they can be added and multiplied by scalars. A *norm* is *any* nonnegative real-valued function $\|\cdot\|$ defined on a linear space such that, for any scalar s and elements x and y of the linear space (vectors or matrices),

$$\begin{aligned} \|x\| &= 0 \text{ iff } x = 0, \\ \|x\| &> 0 \text{ if } x \neq 0, \\ \|sx\| &= |s|\|x\|, \\ \|x + y\| &\leq \|x\| + \|y\|, \end{aligned}$$

where iff stands for "if and only if." These constraints are rather loose, and many possible norms can be defined for a particular linear space. A linear space with a specified norm is called a *normed linear space*. The norm induces a *topology* on the linear space, which is used to define continuity and convergence. Norms are also used in numerical analysis for establishing error bounds and in sensitivity analysis for bounding sensitivities. The multiplicity of norms is useful in these applications, because the user is free to pick the one that works best for her or his particular problem.

We define here many of the more popular norms, some of which are known by more than one name.

B.5.2 Hölder Norms

The inner product of a column n-vector x with itself is

$$\begin{aligned} x^{\mathrm{T}}x &= \text{trace } xx^{\mathrm{T}} \\ &= \sum_{i=1}^{n} x_i^2 \\ &\stackrel{\text{def}}{=} \|x\|_E^2, \end{aligned}$$

the square of the *Euclidean norm* of x. This is but one of a class of norms called *Hölder norms*,[15] ℓ_p *norms*,[16] or simply *p-norms*:

$$\|x\|_p \stackrel{\text{def}}{=} \left[\sum_{i=1}^{n} |x_i|^p \right]^{1/p},$$

[15]Named for the German mathematician Otto Ludwig Hölder (1859–1937).

[16]This "little ℓ "notation is used for infinite-dimensional normed vector spaces (sequences), which include finite-dimensional normed vector spaces as a subclass.

and in the limit (as $p \to \infty$) as the *sup*[17] norm, or ∞-norm:

$$\|x\|_\infty \stackrel{\text{def}}{=} \max_i |x_i|.$$

These norms satisfy the *Hölder inequality*:

$$|x^{\mathrm{T}} y| \leq \|x\|_p \|y\|_q \text{ for } \frac{1}{p} + \frac{1}{q} = 1.$$

They are also related by inequalities such as

$$\|x\|_\infty \leq \|x\|_E \leq \|x\|_1 \leq n\|x\|_\infty.$$

The Euclidean norm (2-norm) is the default norm for vectors. When no other norm is identified, the implied norm is the Euclidean norm.

B.5.3 Matrix Norms

Many norms have been defined for matrices. Two general types are presented here. Both are derived from vector norms, but by different means.

B.5.3.1 Generalized Vector Norms Vector norms can be generalized to matrices by treating the matrix like a doubly subscripted vector. For example, the Hölder norms for vectors can be generalized to matrices as

$$\|A\|_{(p)} = \left\{ \sum_{i=1}^{m} \sum_{j=1}^{n} |a_{i,j}|^p \right\}^{1/p}.$$

The matrix (2)-norm defined in this way is also called the *Euclidean norm, Schur norm*, or *Frobenius norm*. We will use the notation $\|\cdot\|_F$ in place of $\|\cdot\|_{(2)}$ for the Frobenius norm.

The reason for putting parentheses around the subscript p in the above definition is that there is another way that the vector p-norms are used to define matrix norms, and it is this alternative definition that is usually allowed to wear an unadorned p subscript. These alternative norms also have the following desirable properties.

B.5.3.2 Desirable Multiplicative Properties of Matrix Norms Because matrices can be multiplied, one could also apply the additional constraint that

$$\|AB\|_M \leq \|A\|_M \|B\|_M$$

[17] *sup* (sounds like "soup") stands for *supremum*, a mathematical term for the *least upper bound* of a set of real numbers. The maximum (max) is the supremum over a finite set.

for conformable matrices A and B and a matrix norm $\|\cdot\|_M$. This is a good property to have for some applications. One might also insist on a similar property with respect to multiplication by vector x, for which a norm $\|\cdot\|_{V_1}$ may already be defined:

$$\|Ax\|_{V_2} \leq \|A\|_M \|x\|_{V_1}.$$

This property is called *compatibility* between the matrix norm $\|\cdot\|_M$ and the vector norms $\|\cdot\|_{V_1}$ and $\|.\|_{V_2}$. (Note that there can be two distinct vector norms associated with a matrix norm: one in the normed linear space containing x and one in the space containing Ax.)

B.5.3.3 Matrix Norms Subordinate to Vector Hölder Norms There is a family of alternative matrix "p-norms" [but not (p)-norms] defined by the formula

$$\|A\|_p \stackrel{\text{def}}{=} \sup_{\|x\| \neq 0} \frac{\|Ax\|_p}{\|x\|_p},$$

where the norms on the right-hand side are the vector Hölder norms and the induced matrix norms on the left are called *subordinate* to the corresponding Hölder norms. The 2-norm defined in this way is also called the *spectral norm* of A. It has the properties

$$\|\text{diag}_i\{\lambda_i\}\|_2 = \max_i |\lambda_i| \text{ and } \|Ax\|_2 \leq \|A\|_2 \|x\|_2.$$

The first of these properties implies that $\|I\|_2 = 1$. The second property is compatibility between the spectral norm and the vector Euclidean norm. (Subordinate matrix norms are guaranteed to be compatible with the vector norms used to define them.) All matrix norms subordinate to vector norms also have the property that $\|I\| = 1$.

B.5.3.4 Computation of Matrix Hölder Norms The following formulas may be used in computing 1-norms and ∞-norms of $m \times n$ matrices A:

$$\|A\|_1 = \max_{1 \leq j \leq n} \left\{ \sum_{i=1}^{m} |a_{ij}| \right\},$$

$$\|A\|_\infty = \max_{1 \leq i \leq m} \left\{ \sum_{j=1}^{n} |a_{ij}| \right\}.$$

The norm $\|A\|_2$ can be computed as the square root of the largest characteristic value of $A^T A$, which takes considerably more effort.

B.5.3.5 Default Matrix Norm When the type of norm applied to a matrix is not specified (by an appropriate subscript), the default will be the spectral norm (Hölder matrix 2-norm). It satisfies the following bounds with respect to the Frobenius norm and the other matrix Hölder norms for $m \times n$ matrices A:

$$\begin{aligned}
\|A\|_2 &\le \|A\|_F \le \sqrt{n}\|A\|_2, \\
\frac{1}{\sqrt{m}}\|A\|_1 &\le \|A\|_2 \le \sqrt{n}\|A\|_1, \\
\frac{1}{\sqrt{n}}\|A\|_\infty &\le \|A\|_2 \le \sqrt{m}\|A\|_\infty, \\
\max_{\substack{1\le i\le m \\ 1\le j\le n}} |a_{ij}| &\le \|A\|_2 \le \sqrt{mn} \max_{\substack{1\le i\le m \\ 1\le j\le n}} |a_{ij}|.
\end{aligned}$$

B.6 CHOLESKY DECOMPOSITION

This decomposition is named after André Louis Cholesky, a French[18] geodesist and artillery officer and a casualty of World War I. He discovered a method for solving linear least-squares problems that uses a method for factoring a symmetric, positive-definite matrix P as a product of triangular factors. He was perhaps not the first discoverer[19] of the factoring technique, but his use of the method for solving least-squares problems was unique. His results were published posthumously by a fellow officer, Commendant Benoit [30], and credited to Cholesky. Choleksy decomposition is used in several ways for implementing Kalman filters.

B.6.1 Matrix Square Roots and Cholesky Factors

A square root of a matrix M is a matrix S such that $M = S^2 = SS$. The matrix square root is sometimes confused with a Cholesky factor, which is not the same thing. A Cholesky factor of a symmetric positive-definite matrix P is a matrix C such that

$$CC^{\mathrm{T}} = P. \tag{B.5}$$

Note that it does not matter whether we write this equation in the alternative form $F^{\mathrm{T}}F = P$, because the two solutions are related by $F = C^{\mathrm{T}}$.

[18]Because Cholesky was French, his name should perhaps be pronounced "Show-less-KEY," with the accent on the last syllable.

[19]Zurmühl [294] cites an earlier discovery by M. H. Doolittle, published in a U.S. Coast and Geodetic Report in 1878.

B.6.1.1 Cholesky Factors Are Not Unique If C is a Cholesky factor of P, then for any conformable orthogonal matrix M, the matrix

$$A \stackrel{\text{def}}{=} CM \tag{B.6}$$

satisfies the equation

$$\begin{aligned} AA^{\mathrm{T}} &= CM(CM)^{\mathrm{T}} \\ &= CMM^{\mathrm{T}}C^{\mathrm{T}} \\ &= CC^{\mathrm{T}} \\ &= P. \end{aligned}$$

That is, A is also a legitimate Cholesky factor.

The ability to transform one Cholesky factor into another using orthogonal matrices turns out to be very important in square-root filtering (in Section 6.5).

B.6.2 Cholesky Factoring Algorithms

There are two possible forms of Cholesky factorization, corresponding to two possible forms of the defining equation:

$$P = L_1 L_1^{\mathrm{T}} = U_1^{\mathrm{T}} U_1 \tag{B.7}$$

$$= U_2 U_2^{\mathrm{T}} = L_2^{\mathrm{T}} L_2, \tag{B.8}$$

where the Cholesky factors U_1, U_2 are upper triangular and their respective transposes L_1, L_2 are lower triangular.

The first of these is implemented by the built-in MATLAB function `chol(P)`, with argument P a symmetric positive-definite matrix. The call `chol(P)` returns an upper triangular matrix U_1 satisfying Equation B.7. The MATLAB m-file `utchol.m` on the accompanying CD implements the solution to Equation B.8. The call `utchol(P)` returns an upper triangular matrix U_2 satisfying Equation B.8.

There are also two possible forms for each of the two factoring algorithms, depending on whether the second level of the indexing loop is by rows or columns, but this detail has no significant effect on the result.

B.6.3 Modified Cholesky Factorization

The algorithm for Cholesky factorization of a matrix requires taking square roots, which can be avoided by using a *modified Cholesky factorization* in the form

$$P = UDU^{\mathrm{T}}, \tag{B.9}$$

where D is a diagonal matrix with positive diagonal elements and U is a *unit triangular matrix* (i.e., U has 1s along its main diagonal). This algorithm is implemented in the file `modchol.m` on the accompanying CD.

B.6.4 Rank 1 Modifications of Cholesky Factors

A *rank 1 modification* of a Cholesky factor C_0 such that $C_0 C_0^{\mathrm{T}} = A$ is a Cholesky factor of $A \pm vv^{\mathrm{T}}$, where v is a column vector and vv^{T} is a rank 1 matrix. The built-in MATLAB function `cholupdate.m` performs that function, given C_0 and v, for factorization in the form $C_{\text{output}}^{\mathrm{T}} C_{\text{output}} = C_{\text{input}}^{\mathrm{T}} C_{\text{input}} + vv^{\mathrm{T}}$(i.e., transposing the first factor rather than the second factor). The MATLAB rank 1 Cholesky factor modification functions `potter.m, carlson.m,` and `bierman.m` on the accompanying CD are specialized for application to Kalman filtering, as described in Chapter 6.

B.7 ORTHOGONAL DECOMPOSITIONS OF MATRICES

Decompositions are also called *factorizations* of matrices. They are formulas for representing a matrix as a product of matrix factors with useful properties. The two factorizations described here have either triangular or diagonal factors in addition to orthogonal factors.

Decomposition methods are algorithms for computing the factors, given the matrix to be decomposed.

B.7.1 *QR* Decomposition (Triangularization)

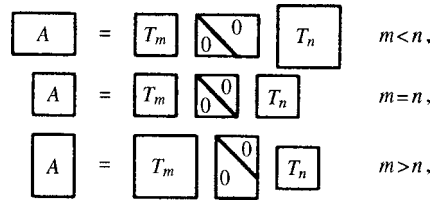
The *QR decomposition* of a matrix A is a representation in the form

$$A = QR,$$

where Q is an orthogonal matrix and R is a triangular matrix. Methods for the *QR* decomposition of a matrix are described in Chapter 6.

B.7.2 Singular-Value Decomposition

The *singular-value decomposition* of an $m \times n$ matrix A is a representation in the form $A = T_m D T_n$, where T_m and T_n are orthogonal matrices (with square dimensions as specified by their subscripts) and D is an $m \times n$ matrix filled with zeros everywhere except along the main diagonal of its maximal upper-left square submatrix. This decomposition will have either of the three forms:



depending on the relative values of m and n. The middle matrix D has the block form

$$D = \begin{cases} [\text{diag}_i\{\sigma_i\} | 0_{m\times(n-m)}] & \text{if } m < n, \\ \text{diag}_i\{\sigma_i\} & \text{if } m = n, \\ \begin{bmatrix} \text{diag}_i\{\sigma_i\} \\ 0_{(m-n)\times n} \end{bmatrix} & \text{if } m > n, \end{cases}$$

$$\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq \cdots \geq \sigma_p \geq 0,$$

$$p = \min(m, n).$$

That is, the diagonal nonzero elements of D are in *descending order* and nonnegative. These are called the *singular values* of A. For a proof that this decomposition exists and an algorithm for computing it, see the book by Golub and Van Loan [99].

The singular values of a matrix characterize many useful matrix properties, such as:

$\|A\|_2 = \sigma_1\ (A)$.

rank $(A) = r$ such that $\sigma_r > 0$ and either $\sigma_{r+1} = 0$ or $r = p$. (The rank of a matrix is defined in Section B.3.2.)

The condition number of A equals σ_1/σ_p.

The condition number of the matrix A in the linear equation $Ax = b$ bounds the sensitivity of the solution x to variations in b and the sensitivity of the solution to roundoff errors in determining it. The singular-value decomposition may also be used to define the *pseudorank* of A as the smallest singular value σ_i such that $\sigma_i > \varepsilon\sigma_1$, where ε is a processor- and precision-dependent constant such that $0 < \varepsilon \ll 1$.

These relationships are useful for the analysis of state transition matrices Φ of Kalman filters, which can be singular or close enough to being singular that numerical roundoff can cause the product $\Phi\ P\ \Phi^{\mathrm{T}}$ to be essentially singular.

B.7.3 Eigenvalue–Eigenvector Decompositions of Symmetric Matrices

Symmetric QR Decomposition The so-called symmetric QR decomposition of an $n \times n$ symmetric real matrix A has the special form $A = TDT^{\mathrm{T}}$, where the right orthogonal matrix is the transposed left orthogonal matrix and the diagonal matrix

$$D = \text{diag}_i\{\lambda_i\}.$$

That is, the diagonal elements are the characteristic values of the symmetric matrix. Furthermore, the column vectors of the orthogonal matrix T are the

associated characteristic vectors e_i of A:

$$\begin{aligned} A &= TDT^{\mathrm{T}} \\ &= \sum_{i=1}^{n} \lambda_i e_i e_i^{\mathrm{T}}, \\ T &= [e_1 \quad e_2 \quad e_3 \quad \cdots \quad e_n]. \end{aligned}$$

These relationships are useful for the analysis of covariance matrices, which are constrained to have nonnegative characteristic values, although their numerical values may stray enough in practice (due to computer roundoff errors) to develop negative characteristic values.

B.8 QUADRATIC FORMS

Bilinear and Quadratic Forms For a matrix A and all conformable column vectors x and y, the functional mapping $(x, y) \rightarrow x^{\mathrm{T}} Ay$ is called a *bilinear form*. As a function of x and y, it is linear in both x and y and hence *bilinear*. In the case where $x = y$, the functional mapping $x \rightarrow x^{\mathrm{T}} Ax$ is called a *quadratic form*. The matrix A of a quadratic form is always a square matrix.

B.8.1 Symmetric Decomposition of Quadratic Forms

Any square matrix A can be represented uniquely as the sum of a symmetric matrix and a skew-symmetric matrix:

$$A = \frac{1}{2}(A + A^{\mathrm{T}}) + \frac{1}{2}(A - A^{\mathrm{T}}),$$

where $\frac{1}{2}(A + A^{\mathrm{T}})$ is called the *symmetric* part of A and $\frac{1}{2}(A - A^{\mathrm{T}})$ is called the *skew-symmetric* or *antisymmetric* part of A. The quadratic form $x^T Ax$ depends only on the symmetric part of A:

$$x^{\mathrm{T}} Ax = x^{\mathrm{T}} \left\{ \frac{1}{2}(A + A^{\mathrm{T}}) \right\} x.$$

Therefore, one can always assume that the matrix of a quadratic form is symmetric, and one can express the quadratic form in summation form as

$$\begin{aligned} x^{\mathrm{T}} Ax &= \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} x_i x_j \\ &= \sum_{i=j} a_{ij} x_i x_j + \sum_{i \neq j} a_{ij} x_i x_j \\ &= \sum_{i=1}^{n} a_{ii} x_i^2 + 2 \sum_{i>j} a_{ij} x_i x_j \end{aligned}$$

for symmetric A.

B.8.1.1 Ranges of Quadratic Forms The domain of a quadratic form for an $n \times n$ matrix is n-dimensional Euclidean space, and the range is in $(-\infty, +\infty)$, the real line. Assume that $x \neq 0$. Then:

If A is positive definite, the range of $x \rightarrow x^T Ax$ is $(0, +\infty)$,
If A is nonnegative-definite, the range of $x \rightarrow x^{\mathrm{T}} Ax$ is $[0, +\infty)$,
If A is indefinite, the range of $x \rightarrow x^{\mathrm{T}} Ax$ is $(-\infty, +\infty)$,
If A is nonpositive-definite, the range of $x \rightarrow x^{\mathrm{T}} Ax$ is $(-\infty, 0]$, and
If A is negative-definite, the range of $x \rightarrow x^{\mathrm{T}} Ax$ is $(-\infty, 0)$.

If $x^{\mathrm{T}} x = 1$, then $\lambda_n(A) \leq x^{\mathrm{T}} Ax \leq \lambda_1(A)$. That is, the quadratic form maps the unit n-sphere onto the closed interval $[\lambda_n(A), \lambda_1(A)]$.

B.9 DERIVATIVES OF MATRICES

B.9.1 Derivatives of Matrix-Valued Functions

The derivative of a matrix with respect to a scalar is the matrix of derivatives of its elements:

$$F(t) = \begin{bmatrix} f_{11}(t) & f_{12}(t) & f_{13}(t) & \cdots & f_{1n}(t) \\ f_{21}(t) & f_{22}(t) & f_{23}(t) & \cdots & f_{2n}(t) \\ f_{31}(t) & f_{32}(t) & f_{33}(t) & \cdots & f_{3n}(t) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ f_{m1}(t) & f_{m2}(t) & f_{m3}(t) & \cdots & f_{mn}(t) \end{bmatrix},$$

$$\frac{d}{dt}F(t) = \begin{bmatrix} \frac{d}{dt}f_{11}(t) & \frac{d}{dt}f_{12}(t) & \frac{d}{dt}f_{13}(t) & \cdots & \frac{d}{dt}f_{1n}(t) \\ \frac{d}{dt}f_{21}(t) & \frac{d}{dt}f_{22}(t) & \frac{d}{dt}f_{23}(t) & \cdots & \frac{d}{dt}f_{2n}(t) \\ \frac{d}{dt}f_{31}(t) & \frac{d}{dt}f_{32}(t) & \frac{d}{dt}f_{33}(t) & \cdots & \frac{d}{dt}f_{3n}(t) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{d}{dt}f_{m1}(t) & \frac{d}{dt}f_{m2}(t) & \frac{d}{dt}f_{m3}(t) & \cdots & \frac{d}{dt}f_{mn}(t) \end{bmatrix}.$$

The rule for the derivative of a product applies also to matrix products:

$$\frac{d}{dt}[A(t)B(t)] = \left[\frac{d}{dt}A(t)\right]B(t) + A(t)\left[\frac{d}{dt}B(t)\right],$$

provided that the order of the factors is preserved. If $F(t)$ is square and nonsingular, then $F(t)F^{-1}(t) = I$, a constant. As a consequence, its derivative will be zero. This

fact can be used to derive the formula for the derivative of a matrix inverse:

$$\begin{aligned} 0 &= \frac{d}{dt} I \\ &= \frac{d}{dt}[F(t)F^{-1}(t)] \\ &= \left[\frac{d}{dt}F(t)\right]F^{-1}(t) + F(t)\left[\frac{d}{dt}F^{-1}(t)\right], \\ \frac{d}{dt}F^{-1}(t) &= -F^{-1}\left[\frac{d}{dt}F(t)\right]F^{-1} \end{aligned} \tag{B.10}$$

B.9.2 Gradients of Quadratic Forms

If $f(x)$ is a differentiable scalar-valued function of an n-vector x, then the vector

$$\frac{\partial f}{\partial x} = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \frac{\partial f}{\partial x_3}, \ldots, \frac{\partial f}{\partial x_n}\right]^{\mathrm{T}}$$

is called the *gradient* of f with respect to x. In the case where f is a quadratic form with symmetric matrix A, the ith component of its gradient will be

$$\begin{aligned} \left[\frac{\partial}{\partial x}(x^{\mathrm{T}}Ax)\right]_i &= \frac{\partial}{\partial x_i}\left(\sum_j a_{jj}x_j^2 + 2\sum_{j<k} a_{jk}x_j x_k\right) \\ &= \left(2a_{ii}x_i + 2\sum_{i<k} a_{ik}x_k + 2\sum_{j<i} a_{ji}x_j\right) \\ &= \left(2a_{ii}x_i + 2\sum_{i\neq k} a_{ik}x_k\right) \\ &= 2\sum_{k=1}^{n} a_{ik}x_k \\ &= (2Ax)_i. \end{aligned}$$

That is, the gradient vector can be expressed as

$$\frac{\partial}{\partial x}(x^{\mathrm{T}}Ax) = 2Ax.$$

BIBLIOGRAPHY

[1] F. S. Acton, *Numerical Methods That Usually Work*, Mathematical Association of America, Washington DC, 1991 (revision of the 1970 edition published by Harper & Rowe, New York).

[2] W. S. Agee and R. H. Turner, *Triangular Decomposition of a Positive Definite Matrix Plus a Symmetric Dyad, with Applications to Kalman Filtering*, White Sands Missile Range Technical Report No. 38, Oct. 1972.

[3] D. W. Allan, "Statistics of atomic frequency standards," *IEEE Proceedings*, Vol. 54, pp. 221–230, 1966.

[4] E. Allen, *Modeling with Itó Stochastic Differential Equations*, Springer, Berlin, 2007.

[5] B. D. O. Anderson, "Properties of optimal linear smoothing," *IEEE Transactions on Automatic Control*, Vol. AC-14, pp. 114–115, 1969.

[6] B. D. O. Anderson, "Second-order convergent algorithms for the steady-state Riccati equation," *International Journal of Control*, Vol. 28, pp. 295–306, 1978.

[7] B. D. O. Anderson and S. Chirarattananon, "Smoothing as an improvment on filtering: A universal bound," *Electronics Letters*, Vol. 7, No. 18, pp. 524–525, 1971.

[8] B. D. O. Anderson and J. B. Moore, *Optimal Filtering*, Prentice-Hall, Englewood Cliffs, NJ, 1979.

[9] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbvaum, S. Hammerling, A. McKenney, S. Ostrouchov, and D. Sorensen, *LAPACK Users' Guide*, Society for Industrial and Applied Mathematics, Philadelphia, 1992.

[10] A. Andrews, "A square root formulation of the Kalman covariance equations," *AIAA Journal*, Vol. 6, pp. 1165–1166, 1968.

Kalman Filtering: Theory and Practice using MATLAB®, Third Edition. By Mohinder S. Grewal and Angus P. Andrews

[11] A. Andrews, "Calibrating the drift rates of strapdown electrostatic gyroscopes," in *IEEE National Aerospace Electronics Conference Proceedings*, pp. 620–627, 1973.

[12] A. Andrews, "Marginal optimization of observation schedules," *AIAA Journal of Guidance and Control*, Vol. 5, pp. 95–96, 1982.

[13] R. N. Ansher, ed., *Physics and Beyond*, Harper & Row, New York, 1971.

[14] ANSI/IEEE Std. 754-1985, *IEEE Standard for Binary Floating-Point Arithmetic*, Institute of Electrical and Electronics Engineers, New York, 1985.

[15] A. C. Antoulas, ed., *Mathematical System Theory: The Influence of R. E. Kalman*, Springer-Verlag, Berlin, 1991.

[16] L. Arnold, *Stochastic Differential Equations: Theory and Applications*, Wiley, New York, 1974.

[17] R. B. Asher, P. S. Maybeck, and R. A. K. Mitchell, "Filtering for precision pointing and tracking with application for aircraft to satellite tracking," in *Proceedings of the IEEE Conference Decision and Control*, Houston, TX, pp. 439–446, 1975.

[18] M. Athans et al., guest eds., Special Issue on Linear-Quadratic-Gaussian Problem, *IEEE Transactions on Automatic Control*, Vol. AC-16, 1971.

[19] R. M. L. Baker and M. W. Makemson, *An Introduction to Astrodynamics*, Academic Press, New York, 1960.

[20] A. V. Balakrishnan, *Kalman Filtering Theory*, Optimization Software, New York, 1987.

[21] J. Baras and V. Mirelli, *Recent Advances in Stochastic Calculus*, Springer-Verlag, New York, 1990.

[22] R. W. Bass, "Some Reminiscences of Control and System Theory in the Period 1955–1960," talk presented at *the 34th Southeastern Symposium on System Theory,* University of Alabama at Huntsville, March 18, 2002.

[23] R. W. Bass, V. D. Norum, and L. Schwartz, "Optimal multichannel nonlinear filtering," *Journal of Mathematical Analysis and Applications*, Vol. 16, pp. 152–164, 1966.

[24] R. H. Battin, *Astronautical Guidance*, McGraw-Hill, New York, 1964.

[25] R. H. Battin, "Space guidance evolution—a personal narrative," *AIAA Journal of Guidance and Control*, Vol. 5, pp. 97–110, 1982.

[26] R. H. Battin, *An Introduction to the Mathematics and Methods of Astrodynamics*, American Institute of Aeronautics and Astronautics, New York, 1987.

[27] J. F. Bellantoni and K. W. Dodge, "A square root formulation of the Kalman–Schmidt filter," *AIAA Journal*, Vol. 5, pp. 1309–1314, 1967.

[28] T. R. Benedict and G. W. Bordner, "Synthesis of an optimal set of radar track-while-scan smoothing equations," *IEEE Transactions on Automatic Control*, Vol. AC-7, pp. 27–32, 1962.

[29] J. M. Bennet, "Triangular factors of modified matrices," *Numerische Mathematik*, Vol. 7, pp. 217–221, 1963.

[30] Commandant Benoit, "Sur une méthode de résolution des équations normales provenant de l'application de la méthode des moindes carrés a un système d'équations linéaires en numbre inférieur a celui des inconnues—application de la méthode a la resolution d'un système defini d'équations linéaires (Procédé du Commandant Cholesky)," *Bulletin Géodêsique et Géophysique Internationale*, Vol. 2, Toulouse, pp. 67–77, 1924.

[31] G. J. Bierman, *Factorization Methods for Discrete Sequential Estimation*, Academic Press, New York, 1977.

[32] G. J. Bierman, "A new computationally efficient fixed-interval discrete time smoother," *Automatica*, Vol. 19, pp. 503–561, 1983.

[33] P. Billingsley, *Probability and Measure*, Wiley, New York, 1986.

[34] K. K. Biswas and A. K. Mahalanabis, "Optimal fixed-lag smoothing for time-delayed systems with colored noise," *IEEE Transactions on Automatic Control*, Vol. AC-17, pp. 387–388, 1972.

[35] K. K. Biswas and A. K. Mahalanabis, "On the stability of a fixed-lag smoother," *IEEE Transactions on Automatic Control*, Vol. AC-18, pp. 63–64, 1973.

[36] K. K. Biswas and A. K. Mahalanabis, "On computational aspects of two recent smoothing algorithms," *IEEE Transactions on Automatic Control*, Vol. AC-18, pp. 395–396, 1973.

[37] S. Bittanti, A. J. Laub, and J. C. Willems, eds., *The Riccati Equation*, Springer-Verlag, New York, 1991.

[38] Å. Björck, "Solving least squares problems by orthogonalization," *BIT*, Vol. 7, pp. 1–21, 1967.

[39] J. H. Blakelock, *Automatic Control of Aircraft and Missiles*, Wiley, New York, 1965.

[40] F. R. Bletzacker, D. H. Eller, T. M. Forgette, G. L. Siebert and J. L. Vavrus, "Kalman filter design for integration of Phase III GPS with an inertial navigation system," *Computing Applications Software Technology Technical Papers*, Los Alamitos, CA, Institute of Navigation, Washington, DC, 1988.

[41] H. W. Bode and C. E. Shannon, "A simplified derivation of linear least-squares smoothing and prediction," *IRE Proceedings*, Vol. 48, pp. 417–425, 1950.

[42] E. Bodewig, *Matrix Calculus*, North-Holland, Amsterdam, 1959.

[43] J. E. Bortz, "A new mathematical formulation for strapdown inertial navigation," *IEEE Transactions of Aerospace and Electronic Systems*, Vol. AES-6, pp. 61–66, 1971.

[44] J.-L. Botto and G. V. Moustakides, "Stabilizing the fast Kalman filter algorithms," *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol. ASSP-37, pp. 1342–1348, 1989.

[45] S. M. Bozic, *Digital and Kalman Filtering: An Introduction to Discrete-Time Filtering and Optimal Linear Estimation*, Wiley, New York, 1979.

[46] K. Brammer and G. Siffling, *Kalman–Bucy Filters*, Artech House, Norwood, MA, 1989.

[47] R. W. Brockett, *Finite Dimensional Linear Systems*, Wiley, New York, 1970.

[48] K. Brodie, D. Eller, and G. Seibert, "Performance analysis of integrated navigation systems," in *Proceedings of the 1985 Winter Simulation Conference* (D. Gantz, G. Glais, and S. Solomon, eds.), pp. 605–609, San Francisco, 1985.

[49] R. G. Brown, *Introduction to Random Signal Analysis and Kalman Filtering*, Wiley, New York, 1983.

[50] R. G. Brown and P. Y. C. Hwang, *Introduction to Random Signals and Applied Kalman Filtering*, 2nd ed., Wiley, New York, 1992.

[51] R. G. Brown and P. Y. C. Hwang, *Introduction to Random Signals and Applied Kalman Filtering: With MATLAB Exercises and Solutions*, 3rd ed., Wiley, New York, 1997.

[52] A. E. Bryson, Jr., and Y.-C. Ho, *Applied Optimal Control*, Blaisdell, Waltham, MA, 1969.

[53] A. E. Bryson, Jr., and D. E. Johansen, "Linear filtering for time-varying systems using measurements containing colored noise," *IEEE Transactions on Automatic Control*, Vol. AC-10, pp. 4–10, 1965.

[54] R. S. Bucy, "Nonlinear filtering theory," *IEEE Transactions on Automatic Control*, Vol. AC-10, pp. 198–206, 1965.

[55] R. S. Bucy, "Optimal filtering for correlated noise," *Journal of Mathematical Analysis Applications*, Vol. 20, pp. 1–8, 1967.

[56] R. S. Bucy and P. D. Joseph, *Filtering for Stochastic Processes, with Applications to Guidance*, Wiley, New York, 1968.

[57] N. A. Carlson, "Fast triangular formulation of the square root filter," *AIAA Journal*, Vol. 11, No. 9, pp. 1259–1265, 1973.

[58] D. E. Catlin, *Estimation, Control, and the Discrete Kalman Filter*, Springer-Verlag, New York, 1989.

[59] J. L. Center, J. A. D'Appolito, and S. I. Marcus, *Reduced-Order Estimators and Their Application to Aircraft Navigation*, Tech. Rep., TASC TR-316-4-2, The Analytic Sciences Corporation, Reading, MA, Aug. 1974.

[60] G. Chen and C. K. Chui, "A modified adaptive Kalman filter for real-time applications," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 27, pp. 149–154, 1991.

[61] H. F. Chen, *Recursive Estimation and Control for Stochastic Systems*, Wiley, New York, 1985.

[62] C. H. Choi and A. J. Laub, "Efficient matrix-valued algorithms for solving stiff differential Riccati equations," *IEEE Transactions on Automatic Control*, Vol. AC-35, pp. 770–776, 1990.

[63] C. Y. Choe and B. D. Tapley, "A new method for propagating the square root covariance in triangular form," *AIAA Journal*, Vol. 13, pp. 681–683, 1975.

[64] C. K. Chui and G. Chen, *Kalman Filtering with Real-Time Applications*, Springer-Verlag, New York, 1987.

[65] K. L. Chung, *A Course in Probability Theory*, Harcourt Brace, New York, 1968.

[66] E. A. Coddington and N. Levinson, *Theory of Ordinary Differential Equations*, McGraw-Hill, New York, 1955.

[67] H. Cox, "On the estimation of state variables and parameters for noisy dynamic systems," *IEEE Transactions on Automatic Control*, Vol. AC-9, pp. 5–12, 1964.

[68] G. Dahlquist and Å. Björck, *Numerical Methods* (N. Anderson, trans.), Prentice-Hall, Englewood Cliffs, NJ, 1974.

[69] W. D. Davenport and W. L. Root, *Random Signals and Noise*, McGraw-Hill, New York, 1958.

[70] M. H. A. Davis, *Linear Estimation and Stochastic Control*, Halsted Press, New York, 1977.

[71] E. J. Davison and M. C. Maki, "The numerical solution of the matrix Riccati differential equation," *IEEE Transactions on Automatic Control*, Vol. AC-19, pp. 71–73, 1973.

[72] P. M. DeRusso, R. J. Roy, and C. M. Close, *State Variables for Engineers*, Wiley, New York, 1965.

[73] L. Dieci, "Numerical integration of the differential Riccati equation and some related issues," *SIAM Journal of Numerical Analysis*, Vol. 29, pp. 781–815, 1992.

[74] R. C. DiPietro and F. A. Farrar, *Comparative Evaluation of Numerical Methods for Solving the Algebraic Matrix Riccati Equation*, Report No. R76-140268-1, United Technologies Research Center, East Hartford, CT, 1976.

[75] J. J. Dongarra, C. B. Moler, J. R. Bunch, and G. W. Stewart, *LINPACK Users' Guide,* Society for Industrial and Applied Mathematics, Philadelphia, 1979.

[76] R. C. Dubes, *The Theory of Applied Probability*, Prentice-Hall, Englewood Cliffs, NJ, 1968.

[77] R. M. DuPlessis, *Poor Man's Explanation of Kalman Filtering, or How I Stopped Worrying and Learned to Love Matrix Inversion*, Rep. No. QN014239, North American Rockwell, Autonetics Division, Anaheim, CA, 1967.

[78] P. Dyer and S. McReynolds, "Extension of square-root filtering to include process noise," *Journal of Optimization Theory and Applications*, Vol. 3, pp. 444–458, 1969.

[79] R. Eckhardt, "Stan Ulam, John von Neumann, and the Monte Carlo method," *Los Alamos Science*, Special Issue, pp. 131–143, 1987.

[80] A. Einstein, "Über die von molekularkinetischen Theorie der Wärme geforderte Bewegung von in ruhenden Flüssigkeiten suspendierten Teilchen," *Annelen der Physik*, Vol. 17, pp. 549–560, 1905.

[81] E. Eskow and R. B. Schnabel, "Algorithm 695: Software for a new modified Cholesky factorization," *Collected Algorithms of the ACM*, Association for Computing Machinery, New York, 1991.

[82] A. F. Fath, "Computational aspects of the linear optimal regulator problem," *IEEE Transactions on Automatic Control*, Vol. AC-14, pp. 547–550, 1969.

[83] R. A. Fisher, *Statistical Methods for Research Workers*, Oliver and Boyd Edinburgh, 1925.

[84] R. J. Fitzgerald, "Divergence of the Kalman filter," *IEEE Transactions on Automatic Control*, Vol. AC-16, pp. 736–743, 1971.

[85] A. D. Fokker, "Die mittlerer Energie rotierender elektrischer Dipole im Strahlungsfeld," *Annelen der Physik*, Vol. 43, pp. 810–820, 1914.

[86] G. E. Forsythe, M. A. Malcolm, and C. B. Moler, *Computer Methods for Mathematical Computations*, Prentice-Hall, Englewood Cliffs, NJ, 1977.

[87] T. E. Fortmann, "A matrix inversion identity," *IEEE Transactions on Automatic Control*, Vol. AC-15, p. 599, 1970.

[88] R. V. Gamkrelidze, ser. ed., *Encyclopedia of Mathematical Sciences,* Vol. 3*: Dynamical Systems III* (V. I. Arnold, ed.), Springer-Verlag, Berlin, 1988.

[89] F. R. Gantmacher, *The Theory of Matrices*, Vol. 1, Chelsea House, New York, 1990.

[90] F. M. Gaston and G. W. Irwin, "Systolic Kalman filtering: An overview," *IEE Proceedings*, Vol. 137, p. 235–244, 1990.

[91] C. F. Gauss, *Abhandlungen zur Methode der kleinsten Quadrate* (German trans. by A. Borsch and P. Simon), P. Stankiewicz, Berlin, 1887.

[92] C. F. Gauss, *Theory of Motion of the Heavenly Bodies* (English trans.), Dover Publications, New York, 1963.

[93] W. C. Gear, *Numerical Initial Value Problems in Ordinary Differential Equations*, Prentice-Hall, Englewood Cliffs, NJ, 1971.

[94] A. Gelb, "Dual contributions of optimal estimation theory in aerospace applications," *IEEE Control Systems Magazine*, Vol. 6, No. 1, pp. 3–13, 1986.

[95] A. Gelb, J. F. Kasper, Jr., R. A. Nash, Jr., C. F. Price, and A. A. Sutherland, Jr., *Applied Optimal Estimation*, MIT Press, Cambridge, MA, 1974.

[96] W. M. Gentleman, "Least squares computations by Givens transformations without square roots," *Journal of the Institute for Mathematical Applications*, Vol. 12, pp. 329–336, 1973.

[97] W. Givens, "Computation of plane unitary rotations transforming a general matrix to triangular form," *Journal of the Society for Industrial and Applied Mathematics*, Vol. 6, pp. 26–50, 1958.

[98] G. H. Golub, "Numerical methods for solving linear least squares problems," *Numerische Mathematik*, Vol. 7, pp. 206–216, 1965.

[99] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 2nd ed., Johns Hopkins University Press, Baltimore, 1989.

[100] M. S. Grewal, "Application of Kalman filtering to the calibration and alignment of inertial navigation systems," in *Proceedings of PLANS '86—Position Location and Navigation Symposium,* Las Vegas, NV, Nov. 4–7, 1986, IEEE, New York, 1986.

[101] M. S. Grewal and A. P. Andrews, *Application of Kalman Filtering to GPS, INS, and Navigation (Notes)*, Kalman Filtering Consulting Associates, Anaheim, CA, 2000.

[102] M. S. Grewal, V. D. Henderson, and R. S. Miyasako, "Application of Kalman filtering to the calibration and alignment of inertial navigation systems," *IEEE Transactions on Automatic Control*, Vol. AC-38, pp. 4–13, 1991.

[103] M. S. Grewal and R. S. Miyasako, "Gyro compliance estimation in the calibration and alignment of inertial measurement units," in *Proceedings of the Eighteenth Joint Services Conference on Data Exchange for Inertial Systems*, San Diego, CA, Oct. 28–30, 1986, IEEE Joint Services Data Exchange, San Diego, CA, 1986.

[104] M. S. Grewal, R. S. Miyasako, and J. M. Smith, "Application of fixed point smoothing to the calibration, alignment, and navigation data of inertial navigation systems," in *Proceedings of IEEE PLANS '88—Position Location and Navigation Symposium*, Orlando, FL, Nov. 29–Dec. 2, 1988, pp. 476–479, IEEE, New York, 1988.

[105] M. S. Grewal and H. J. Payne, "Identification of parameters in a freeway traffic model," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-6, pp. 176–185, 1976.

[106] M. S. Grewal, L. R. Weill, and A. P. Andrews, *Global Positioning Systems, Inertial Navigation and Integration*, 2nd ed., Wiley, New York, 2007.

[107] C. M. Grinstead and J. L. Snell, *Introduction to Probability*, 2nd ed., American Mathematical Society, Providence, RI, 1997.

[108] B. Harris, *Theory of Probability*, Addison-Wesley, Reading, MA, 1966.

[109] H. Heffes, "The effects of erroneous models on the Kalman filter response," *IEEE Transactions on Automatic Control*, Vol. AC-11, pp. 541–543, 1966.

[110] H. V. Henderson and S. R. Searle, "On deriving the inverse of a sum of matrices," *SIAM Review*, Vol. 23, No. 1, pp. 53–60, Jan. 1981.

[111] C. Hide, T. Moore, C. Hill, and D. Park, "Low cost, high accuracy positioning in urban environments," *Journal of Navigation*, Vol. 59, pp. 365–379, Cambridge University Press, 2006.

[112] A. S. Householder, "Unitary triangularization of a nonsymmetric matrix," *Journal of the Association for Computing Machinery*, Vol. 5, pp. 339–342, 1958.

[113] J. R. Huddle and D. A. Wismer, "Degradation of linear filter performance due to modeling error," *IEEE Transactions on Automatic Control*, Vol. AC-13, pp. 421–423, 1968.

[114] K. Itô, *Lectures on Stochastic Processes*, Tata Institute of Fundamental Research, Bombay (Mumbai), India, 1961.

[115] K. Itô and H. P. McKean, Jr., *Diffusion Processes and Their Sample Paths*, Academic Press, New York, 1965.

[116] A. H. Jazwinski, *Stochastic Processes and Filtering Theory*, Academic Press, New York, 1970.

[117] G. M. Jenkins and D. G. Watts, *Spectral Analysis and Its Applications*, Holden-Day, San Francisco, 1968.

[118] T. L. Jordan, "Experiments on error growth associated with some linear least-squares procedures," *Mathematics of Computation*, Vol. 22, pp. 579–588, 1968.

[119] J. M. Jover and T. Kailath, "A parallel architecture for Kalman filter measurement update and parameter update," *Automatica*, Vol. 22, pp. 783–786, 1986.

[120] S. J. Julier, Comprehensive process models for high-speed navigation, Ph.D. thesis, Oxford University, Oct. 1997.

[121] S. J. Julier, "The spherical simplex unscented transformation," *Proceedings of the 2003 American Control Conference*, Vol. 3, pp. 2430–2434, 2003.

[122] S. J. Julier and J. K. Uhlmann, "A general method for approximating nonlinear transformations of probability distributions," Technical Report, Robotics Research Group, Department of Engineering Science, University of Oxford, 1994.

[123] S. J. Julier and J. K. Uhlmann, "A new extension of the Kalman filter to nonlinear systems," *Proceedings of AeroSense: The 11th International Symposium on Aerospace/Defense Sensing, Simulation and Controls, Multi Sensor Fusion, Tracking and Resource Management*, SPIE, Orlando, FL, April 21–24, pp. 182–193, 1997.

[124] S. J. Julier and J. K. Uhlmann, "Reduced sigma point filters for the propagation of means and covariances through nonlinear transformations," *Proceedings of the IEEE American Control Conference*, Vol. 2, pp. 887–892, 2002.

[125] S. J. Julier and J. K. Uhlmann, "Comment on 'a new method for the nonlinear transformation of means and covariances in filters and estimators' [author's reply]," *IEEE Transactions on Automatic Control*, Vol. 47, No. 8, pp. 1408–1409, 2002.

[126] S. J. Julier, J. K. Uhlmann, and H. F. Durrant-Whyte, "A new approach for filtering nonlinear systems," *Proceedings of the 1995 American Control Conference,* Seattle, WA, pp. 1628–1632, 1995.

[127] S. J. Julier, J. K. Uhlmann, and H. F. Durrant-Whyte, "A new approach for the nonlinear transformation of means and covariances in linear filters," *IEEE Transactions on Automatic Control*, Vol. 45, No. 3, pp. 477–482, 2000.

[128] D. Kahaner, C. Moler, and S. Nash, *Numerical Methods and Software*, Prentice-Hall, Englewood Cliffs, NJ, 1989.

[129] T. Kailath, "An innovations approach to least squares estimation, part I: Linear filtering in additive white noise," *IEEE Transactions on Automatic Control*, Vol. AC-13, pp. 646–655, 1968.

[130] T. Kailath, "A view of three decades of linear filtering theory," *IEEE Transactions on Information Theory*, Vol. IT-20, No. 2, pp. 146–181, 1974.

[131] T. Kailath, "Supplement to 'A Survey of Data Smoothing for Linear and Nonlinear Dynamic Systems'," *Automatica*, Vol. 11, pp. 109–111, 1975.

[132] T. Kailath, "Correspondence item," *Automatica*, Vol. 11, pp. 109–111, 1975.

[133] T. Kailath, *Linear Least Squares Estimation*, Dowden, Hutchinson and Ross, Stroudsburg, PA, 1977.

[134] T. Kailath, *Linear Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1980.

[135] T. Kailath, *Lectures on Weiner and Kalman Filtering*, Springer-Verlag, New York, 1981.

[136] T. Kailath, "Equations of Wiener–Hopf type in filtering theory and related applications," in *Norbert Weiner: Collected Works* (P. Masani, ed.), Vol. 3, MIT Press, Cambridge, MA, 1981.

[137] T. Kailath, "State-space modelling: Square root algorithms," in *Systems and Control Encyclopedia* (M. G. Singh, ed.), Pergamon, Elmsford, NY, 1984.

[138] T. Kailath, "Displacement structure: Theory and applications," *SIAM Review*, Vol. 37, No. 3, pp. 297–386, 1995.

[139] T. Kailath, A. Sayed, and B. Hassibi, *Linear Estimation*, Prentice Hall, Englewood Cliffs, NJ, 2000.

[140] T. Kailath and P. A. Frost, "An innovations approach to least squares estimation, part II: Linear smoothing in additive white noise," *IEEE Transactions on Automatic Control*, Vol. AC-13, pp. 646–655, 1968.

[141] T. Kailath and R. A. Geesey, "An innovations approach to least squares estimation, part IV: Recursive estimation given lumped covariance functions," *IEEE Transactions on Automatic Control*, Vol. AC-16, pp. 720–726, 1971.

[142] R. E. Kalman, Phase-plane analysis of nonlinear sampled-data servomechanisms, S.M. thesis, Department of Electrical Engineering, Massachusetts Institute of Technology, Cambridge, MA, 1954.

[143] R. E. Kalman, "A new approach to linear filtering and prediction problems," *ASME Journal of Basic Engineering*, Vol. 82, pp. 34–45, 1960.

[144] R. E. Kalman, "New methods and results in linear prediction and filtering theory," pp. 270–388 in *Proceedings of the Symposium on Engineering Applications of Random Function Theory and Probability*, J. L. Bogdanoff and F. Kozin, eds., Wiley, New York, 1961.

[145] R. E. Kalman, "New methods in Wiener filtering," pp. 270–388 in *Proceeding of the First Symposium on Engineering Applications of Random Function Theory and Probability*, J. L. Bogdanoff and F. Kozin, eds., Wiley, New York, 1963.

[146] R. E. Kalman and Richard S. Bucy, "New results in linear filtering and prediction theory," *ASME Journal of Basic Engineering*, Series D, Vol. 83, pp. 95–108, 1961.

[147] P. G. Kaminski, Square root filtering and smoothing for discrete processes, Ph.D. thesis, Stanford University, 1971.

[148] P. G. Kaminski, A. E. Bryson, Jr., and S. F. Schmidt, "Discrete square root filtering: A survey of current techniques," *IEEE Transactions on Automatic Control*, Vol. AC-16, pp. 727–736, 1971.

[149] M. H. Kao and D. H. Eller, "Multiconfiguration Kalman filter design for high-performance GPS navigation," *IEEE Transactions on Automatic Control*, Vol. AC-28, pp. 304–314, 1983.

[150] I. Karatzas and S. Shreve, *Brownian Motion and Stochastic Calculus*, Springer-Verlag, New York, 1991.

[151] C. N. Kelly and B. D. O. Anderson, "On the stability of fixed-lag smoothing algorithms," *Journal of the Franklin Institute*, Vol. 291, No. 4, pp. 271–281, 1971.

[152] C. S. Kenney and R. B. Liepnik, "Numerical integration of the differential matrix Riccati equation," *IEEE Transactions on Automatic Control*, Vol. AC-30, pp. 962–970, 1985.

[153] D. W. Klein, "Navigation software design for the user segment of the NAVSTAR GPS," paper presented at the AIAA Guidance and Control Conference, San Diego, CA, Aug. 1976.

[154] A. A. Kolmogorov, "Über die analytichen Methoden in der Wahrscheinlichkeits rechnung," *Mathematische Annelen*, Vol. 104, pp. 415–458, 1931.

[155] M. M. Konstantinov and G. B. Pelova, "Sensitivity of the solutions to differential matrix Riccati equations," *IEEE Transactions on Automatic Control*, Vol. AC-36, pp. 213–215, 1991.

[156] E. Kreindler and P. E. Sarachik, "On the concepts of controllability and observability of linear systems," *IEEE Transactions on Automatic Control*, Vol. AC-9, pp. 129–136, 1964.

[157] L. R. Kruczynski, "Aircraft navigation with the limited operational phase of the Navstar Global Positioning System," *Journal of the Institute of Navigation*, Vol. 1, Global Positioning System: Papers Published in *Navigation*, Institute of Navigation, Alexandria, VA, 1980.

[158] H. J. Kushner, "On the differential equations satisfied by conditional probability densities of Markov processes," *SIAM Journal on Control*, Ser. A Vol. 2, pp. 106–119, 1964.

[159] H. Kwaknernaak and R. Sivan, *Linear Optimal Control Systems*, Wiley, New York, 1972.

[160] D. G. Lainiotis, "Estimation: A brief survey," *Information Sciences*, Vol. 7, pp. 191–202, 1974.

[161] V. Lakshmikantham and A. S. Vatsala, *Generalized Quasilinearization for Nonlinear Problems*, Kluwer Academic Publishers, Norwell, MA, 1998.

[162] P. Langevin, "Sùr la théory du muovement brownien," *Comptes Rendus de l'Academie des Science*, Vol. 146, pp. 530–533, 1908.

[163] J. H. Laning, Jr., and R. H. Battin, *Random Processes in Automatic Control*, McGraw-Hill, New York, 1956.

[164] A. J. Laub, "A Schur method for solving algebraic Riccati equations," *IEEE Transactions on Automatic Control*, Vol. AC-19, pp. 913–921, 1979.

[165] S. L. Lauritzen, "Time series analysis in 1880: A discussion of contributions made by T.N. Thiele," *International Statistical Review*, Vol. 49, No. 3, pp. 319–331, 1981.

[166] A. Lawrence, *Modern Inertial Technology: Navigation, Guidance, and Control*, 2nd ed., Springer-Verlag, New York, 1993.

[167] C. L. Lawson and R. J. Hanson, *Solving Least Squares Problems*, Prentice-Hall, Englewood Cliffs, NJ, 1974.

[168] T. Lefebvre, H. Bruyninckx, and J. De Schutter, "Kalman filters for nonlinear systems: A comparison of performance," *International Journal of Control*, Vol. 77, No. 7, pp. 639–653, 2004.

[169] A. M. Legendre, "Methode de moindres quarres, pour trouver le milieu de plus probable entre les resultats des differentes obvservations," *Memoires Institute de France*, pp. 149–154, 1810.

[170] C. T. Leondes, ed., *Theory and Applications of Kalman Filtering*, AGARDograph No. 139, NATO Advisory Group for Aerospace Research and Development, London, Feb. 1970.

[171] C. T. Leondes, ed., *Advances in the Techniques and Technology of the Application of Nonlinear Filters and Kalman Filters*, AGARDograph No. 256, NATO Advisory Group for Aerospace Research and Development, Paris, 1982.

[172] C. T. Leondes, ed., *Control and Dynamic Systems*, Vols. 19–21: *Nonlinear and Kalman Filtering Techniques*, Parts 1–3, Academic Press, New York, 1983–1984.

[173] N. Levinson, "Wiener's life," *Bulletin of the American Mathematical Society*, Vol. 72, No. 1, Pt. II, pp. 1–32, 1966.

[174] L. J. Levy, "The Kalman filter: Navigation's integration workhorse," *GPS World*, Vol. 8, No. 9, pp. 65–71, 1997.

[175] F. H. Lewis, *Optimal Estimation with an Introduction to Stochastic Control Theory*, Wiley, New York, 1986.

[176] L. Ljung, "Asymptotic behavior of the extended Kalman filter as a parameter estimator for linear systems," *IEEE Transactions on Automatic Control*, Vol. AC-24, pp. 36–50, 1979.

[177] M. Loève, *Probability Theory*, 3rd ed., Van Nostrand Reinhold, New York, 1963.

[178] Y. L. Luke, *The Special Functions and Their Approximations*, Vol. 2, Academic Press, New York, 1969.

[179] A. G. J. MacFarlane, "An eigenvector solution of the optimal linear regulator," *Journal of Electronic Control*, Vol. 14, pp. 643–654, 1963.

[180] P. Masani, ed., *Norbert Weiner: Collected Works*, Vols. 1–4, MIT Press, Cambridge, MA, 1976, 1979, 1981, 1985.

[181] P. Masani, "The life and work of Norbert Wiener," in *Norbert Wiener: Collected Works*, Vol. 4, MIT Press, Cambridge, MA, 1985.

[182] G. Matchett, "GPS-aided shuttle navigation," paper presented at the IEEE National Aerospace and Electronics Conference, 1978.

[183] P. S. Maybeck, *Stochastic Models, Estimation, and Control*, Vol. 1, Academic Press, New York, 1979.

[184] P. S. Maybeck, *Stochastic Models, Estimation, and Control*, Vol. 2, Academic Press, New York, 1982.

[185] P. S. Maybeck, J. G. Reid, and R. N. Lutter, "Application of an extended Kalman filter to an advanced fire control system," in *Proceedings of the IEEE Conference on Decision and Control,* New Orleans, pp. 1192–1195, 1977.

[186] L. A. McGee and S. F. Schmidt, *Discovery of the Kalman Filter as a Practical Tool for Aerospace and Industry*, Technical Memorandum 86847, National Aeronautics and Space Administration, Mountain View, CA, Nov. 1985.

[187] S. R. McReynolds, "Fixed interval smoothing: Revisited," *AIAA Journal of Guidance, Control, and Dynamics*, Vol. 13, pp. 913–921, 1990.

[188] J. S. Meditch, "A survey of data smoothing for linear and nonlinear dynamic systems," *Automatica*, Vol. 9, pp. 151–162, 1973.

[189] R. K. Mehra, "A comparison of several nonlinear filters for reentry vehicle tracking," *IEEE Transactions on Automatic Control*, Vol. AC-16, pp. 307–319, 1971.

[190] J. L. Melsa and D. L. Cohen, *Decision and Estimation Theory*, McGraw-Hill, New York, 1978.

[191] J. M. Mendel, *Discrete Techniques of Parameter Estimation: The Equation Error Formulation*, Marcel Dekker, New York, 1973.

[192] J. M. Mendel, *Lessons in Digital Estimation Techniques*, Prentice-Hall, Englewood Cliffs, NJ, 1987.

[193] J. M. Mendel, "Kalman filtering and other digital estimation techniques," *IEEE Individual Learning Package*, IEEE, New York, 1987.

[194] J. M. Mendel and D. L. Geiseking, "Bibliography on the linear-quadratic-Gaussian problem," *IEEE Transactions on Automatic Control*, Vol. AC-16, pp. 847–869, 1971.

[195] N. Metropolis and S. Ulam, "The Monte Carlo method," *Journal of the American Statistical Association*, Vol. 44, pp. 335–341, 1949.

[196] K. S. Miller, *Some Eclectic Matrix Theory*, Krieger, Malabar, FL, 1987.

[197] C. Moler and C. Van Loan, "Nineteen dubious ways to compute the exponential of a matrix," *SIAM Review*, Vol. 20, pp. 801–836, 1978.

[198] J. B. Moore, "Discrete-time fixed-lag smoothing algorithms," *Automatica*, Vol. 9, No. 2, pp. 163–174, 1973.

[199] J. B. Moore, "Fixed-lag smoothing results for linear dynamical systems," *Australian Telecommunications Research*, Vol. 7, No. 2, pp. 16–21, 1973.

[200] J. B. Moore and K. L. Teo, "Smoothing as an improvement on filtering in high noise," *System & Control Letters*, Vol. 8, pp. 51–54, 1986.

[201] M. Morf and T. Kailath, "Square root algorithms for least squares estimation," *IEEE Transactions on Automatic Control*, Vol. AC-20, pp. 487–497, 1975.

[202] N. E. Nahi, *Estimation Theory and Applications*, Wiley, New York, 1969; reprinted by Krieger, Malabar, FL, 1975.

[203] J. C. Nash, *Compact Numerical Methods for Computers: Linear Algebra and Function Minimization*, Wiley, New York, 1979.

[204] M. B. Nevelson and R. Z. Hazminskii, *Stochastic Approximation and Recursive Estimation*, American Mathematical Society, Translations of Mathematical Monographs, Vol. 47, Providence, RI, 1973.

[205] K. Ogata, *State Space Analysis of Control Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1967.

[206] B. Oksendal, *Stochastic Differential Equations: An Introduction with Applications*, Springer, Berlin, 2007.

[207] H. Padé, "Sur la réprésentation approchée d'une fonction par des fractions rationelle," *Annals d'écoles*, Vol. 9s Suppl., 1892.

[208] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, McGraw-Hill, New York, 1988.

[209] P. G. Park and T. Kailath, "Square-root Bryson–Fraser smoothing algorithms," *IEEE Transactions on Automatic Control*, Vol. 40, pp. 761–766, 1995.

[210] P. G. Park and T. Kailath, "New square-root algorithms for Kalman filtering," *IEEE Transactions on Automatic Control*, Vol. 40, pp. 895–899, 1995.

[211] P. G. Park and T. Kailath, "New square-root smoothing algorithms," *IEEE Transactions on Automatic Control*, Vol. 41, pp. 727–732, 1996.

[212] E. Parzen, *Stochastic Processes*, Holden-Day, San Francisco, 1962.

[213] J. C. Pinson, "Inertial guidance for cruise vehicles," Chapter 4 in *Guidance and Control of Aerospace Vehicles* (C. T. Leondes, ed.), McGraw-Hill, New York, 1963.

[214] M. Planck, "Über einen Satz der statistischen Dynamik und seine Erweiterung in der Quantentheorie," *Sitzungsberichte d. König. Preussischen Akademie der Wissenschaft*, Vol. XXIV, pp. 324–341, 1917.

[215] J. E. Potter, *A Matrix Equation Arising in Statistical Estimation Theory*, Report No. CR-270, National Aeronautics and Space Administration, New York, 1965.

[216] J. E. Potter, "Matrix quadratic solutions," *SIAM Journal of Applied Mathematics*, Vol. 14, pp. 496–501, 1966.

[217] J. E. Potter and R. G. Stern, "Statistical filtering of space navigation measurements," in *Proceedings of the 1963 AIAA Guidance and Control Conference*, AIAA, New York, 1963.

[218] M.-A. Poubelle, I. R. Petersen, M. R. Gevers, and R. R. Bitmead, "A miscellany of results on an equation of Count J. F. Riccati," *IEEE Transactions on Automatic Control*, Vol. AC-31, pp. 651–654, 1986.

[219] S. Prasad and A. K. Mahalanabis, "Finite lag receivers for analog communication." *IEEE Transactions on Communications*, Vol. 23, pp. 204–213, 1975.

[220] R. Premier and A. G. Vacroux, "On smoothing in linear discrete systems with time delays," *International Journal on Control*, Vol. 13, pp. 299–303, 1971.

[221] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vettering, *Numerical Recipes in C++: The Art of Scientific Computing*, Cambridge University Press, Cambridge, 2007.

[222] C. F. Price and R. S. Warren, "An analysis of the divergence problem in the Kalman filter," *IEEE Transactions on Automatic Control*, Vol. AC-13, pp. 699–702, 1968.

[223] M. L. Psiaki, "Square-root information filtering and fixed-interval smoothing with singularities," *Proceedings of the 1998 American Control Conference*, Vol. 5, pp. 2744–2748, Institute for Electrical and Electronics Engineers, Piscataway, NJ. 1988.

[224] A. Ralston and P. Rabinowitz, *A First Course in Numerical Analysis*, McGraw-Hill, New York, 1978.

[225] J. M. Rankin, Kalman filtering approach to market price forecasting, Ph.D. thesis, Iowa State University, Ames, 1986.

[226] M. M. Rao, "Probability," in R. Meyers, ed., *Encyclopedia of Physical Science and Technology*, Academic Press, New York, 1987.

[227] H. E. Rauch, "Solutions to the linear smoothing problem," *IEEE Transactions on Automatic Control*, Vol. AC-8, pp. 371–372, 1963.

[228] H. E. Rauch, F. Tung, and C. T. Striebel, "Maximum likelihood estimates of linear dynamic systems," *AIAA Journal*, Vol. 3, pp. 1445–1450, 1965.

[229] W. T. Reid, *Riccati Differential Equations*, Academic Press, New York, 1972.

[230] J. F. Riccati, "Animadversationnes in aequationes differentiales secundi gradus," *Acta Eruditorum Quae Lipside Publicantur Supplementa*, Vol. 8, pp. 66–73, 1724.

[231] J. M. Richardson and K. A. Marsh, "Nonlinear filtering theory and its applications," *Rockwell International First Annual Signal Processing Conference Proceedings*, 1988, pp. 266–279, Rockwell Science Center, Thousand Oaks, CA.

[232] J. M. Richardson and K. A. Marsh, "Point process theory and the surveillance of many objects," C. R. Smith, G. J. Erikson and P. O. Neudorfer, eds., *Maximum Entropy and Bayesian Methods*, pp. 213–220, Seattle. Kluwer Academic Publishers, Norwell, MA, 1991.

[233] R. M. Rogers, *Applied Mathematics in Integrated Navigation Systems*, 3rd ed., American Institute of Aeronautics and Astronautics, New York, 2007.

[234] P. A. Ruymgaart and T. T. Soong, *Mathematics of Kalman-Bucy Filtering*, Springer-Verlag, Berlin, 1988.

[235] P. G. Savage, *Strapdown Analytics* (2 vols.), Strapdown Associates, Maple Plain, MN, 2000.

[236] F. H. Schlee, C. J. Standish, and N. F. Toda, "Divergence in the Kalman filter," *AIAA Journal*, Vol. 5, pp. 1114–1120, 1967.

[237] G. T. Schmidt, "Linear and nonlinear filtering techniques," in *Control and Dynamic Systems* (C. T. Leondes, ed.), pp. 63–98, Vol. 12, Academic Press, New York, 1976.

[238] G. T. Schmidt, ed., *Practical Aspects of Kalman Filtering Implementation*, AGARD–LS–82, NATO Advisory Group for Aerospace Research and Development, London, May 1976.

[239] S. F. Schmidt, "Applications of state-space methods to navigation problems," in *Advances in Control Systems*, (C. T. Leondes, ed.), Vol. 3, pp. 293–340, Academic Press, New York, 1966.

[240] S. F. Schmidt, "Computational techniques in Kalman filtering," in *Theory and Applications of Kalman Filtering*, AGARDograph 139, NATO Advisory Group for Aerospace Research and Development, London, Feb. 1970.

[241] S. F. Schmidt, "The Kalman filter: Its recognition and development for aerospace applications," *AIAA Journal of Guidance and Control*, Vol. 4, No. 1, pp. 4–8, 1981.

[242] Schuler, M., "Die Storung von Pendul-und Krieselapparaten durch die Beschleunigung der Fahrzeuges," *Physicalische Zeitschrift*, Vol. B, p. 24, 1923.

[243] D. G. Schultz and J. L. Melsa, *State Functions and Linear Control Systems*, McGraw-Hill, New York, 1967.

[244] M. Schwartz and L. Shaw, *Signal Processing, Discrete Spectral Analysis, Detection, and Estimation*, McGraw-Hill, New York, 1975.

[245] F. C. Schweppe, "Evaluation of likelihood functions for Gaussian signals," *IEEE Transactions on Information Theory*, Vol. IT-11, pp. 61–70, 1965.

[246] F. C. Schweppe, *Uncertain Dynamic Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1973.

[247] J. Sherman and W. J. Morrison, "Adjustment of an inverse matrix corresponding to a change in one element of a given matrix," *Annals of Mathematical Statistics*, Vol. 21, pp. 124–127, 1950.

[248] R. A. Singer, "Estimating optimal tracking filter performance for manned maneuvering targets," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. AES-6, pp. 473–483, 1970.

[249] R. A. Singer and K. W. Behnke, "Real-time tracking filter evaluation and selection for tactical applications," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. AES-7, pp. 100–110, 1971.

[250] D. Slepian, "Estimation of signal parameters in the presence of noise," *IRE Transactions on Information Theory*, Vol. IT-3, pp. 68–69, 1954.

[251] K. Sobczyk, *Stochastic Differential Equations with Applications to Physics and Engineering*, Kluwer Academic, Dordrechtx, The Netherlands, 1991.

[252] H. W. Sorenson, "Kalman filtering techniques," in *Advances in Control Systems*, C. T. Leondes, ed., Vol. 3, Academic Press, New York, pp. 219–292, 1966.

[253] H. W. Sorenson, "On the error behavior in linear minimum variance estimation problems," *IEEE Transactions on Automatic Control*, Vol. AC-12, pp. 557–562, 1967.

[254] H. W. Sorenson, "Least-squares estimation: From Gauss to Kalman," *IEEE Spectrum*, Vol. 7, pp. 63–68, 1970.

[255] H. W. Sorenson, "On the development of practical nonlinear filters," *Information Sciences*, Vol. 7, pp. 253–270, 1974.

[256] H. W. Sorenson, ed., *Kalman Filtering: Theory and Application*, IEEE Press, New York, 1985.

[257] R. F. Stengel, *Stochastic Optimal Control: Theory and Application*, Wiley, New York, 1986.

[258] J. Stoer and R. Bulirsh, *Introduction to Numerical Analysis*, Springer-Verlag, New York, 1980.

[259] L. Strachey, *Eminent Victorians*, Penguin Books, London, 1988.

[260] V. Strassen, "Gaussian elimination is not optimal," *Numerische Matematik*, Vol. 13 p. 354, 1969.

[261] R. L. Stratovovich, "Application of the theory of Markoff processes in optimal signal discrimination," *Radio Engineering and Electronic Physics*, Vol. 1, pp. 1–19, 1960.

[262] R. L. Stratonovich, *Topics in the Theory of Random Noise* (R. A. Silverman, ed.), Gordon & Breach, New York, 1963.

[263] D. J. Struik, *A Concise History of Mathematics*, Dover Press, New York, 1987.

[264] S. I. Sudharsanan and M. K. Sundhareshan, "Neural network computational algorithms for least squares estimation problems," in *Proceedings of a Joint Conference on Neural Networks,* Washington DC, IEEE, New York, June 1989, pp. 757–762.

[265] P. Swerling, "First order error propagation in a stagewise differential smoothing procedure for satellite observations," *Journal of Astronautical Sciences*, Vol. 6, pp. 46–52, 1959.

[266] P. Tam and J. B. Moore, "Stable realization of fixed-lag smoothing equations for continuous-time signals," *IEEE Transactions on Automatic Control*, Vol AC-19, No. 1, pp. 84–87, 1974.

[267] T. N. Thiele, "Om Anvendelse af mindste Kvadraters Methode i nogle Tilfälde, hvor en Komplikation af visse Slags uensartede tilfäldige Fejlkilder giver Fejlene en systematisk Karakter," *Det kongelige danske Videnskabernes Selskabs Skrifter, 5 Räkke, naturvidenskabelig og mathematisk Afdeling*, Vol. 12, pp. 381–408, 1880.

[268] C. L. Thornton, Triangular covariance factorizations for Kalman filtering, Ph.D. thesis, University of California at Los Angeles, 1976.

[269] C. L. Thornton and G. J. Bierman, *A Numerical Comparison of Discrete Kalman Filtering Algorithms: An Orbit Determination Case Study*, JPL Technical Memorandum 33-771, NASA/JPL, Pasadena, CA, 1976.

[270] D. H. Titterton and J. L. Weston, *Strapdown Inertial Navigation Technology*, Peter Peregrinus, Stevenage, United Kingdom, 1997.

[271] J. B. Y. Tsui, *Fundamentals of Global Positioning System Receivers: A Software Approach*, 2nd ed., Wiley, New York, 2004.

[272] S. M. Ulam, *Adventures of a Mathematician*, University of California Press, Berkeley, CA, 1991.

[273] T. M. Upadhyay and J. G. Damoulakis, "Sequential piecewise recursive filter for GPS low dynamics navigation," *IEEE Transactions on Aerospace Electronics Systems*, Vol. AES-16, pp. 481–491, 1980.

[274] S. Van Huffel and J. Vanderwalle, *The Total Least Squares Problem: Computational Aspects and Analysis*, Society for Industrial and Applied Mathematics, Philadelphia, 1991.

[275] R. van der Merwe, E. A. Wan, and S. I. Julier, "Sigma-point Kalman filters for nonlinear estimation and sensor-fusion—Applications to integrated navigation," *Proceedings of the AIAA Guidance, Navigation, and Control Conference,* Providence, RI, Aug 2004. American Institute of Aeronautics and Astronautics, New York.

[276] C. F. Van Loan, "Computing integrals involving the matrix exponential," *IEEE Transactions on Automatic Control*, Vol. AC-23, No. 3, pp. 395–404, 1978.

[277] M. Vanbegin, P. Van Dooren, and M. Verhaegen, "Algorithm 675: FORTRAN subroutines for computing the square root covariance filter and square root information filter in dense or Hessenberg forms," *ACM Transactions on Mathematical Software*, Vol. 15, pp. 243–256, 1988.

[278] M. Verhaegen and P. Van Dooren, "Numerical aspects of different Kalman filter implementations," *IEEE Transactions on Automatic Control*, Vol. AC-31, pp. 907–917, 1986.

[279] E. A. Wan and R. van der Merwe, "The unscented Kalman filter," in *Kalman Filtering and Neural Networks* (S. Hayken, ed.), Wiley, New York, pp. 221–280, 2001.

[280] R. C. Ward, "Numerical computation of the matrix exponential with accuracy estimate," *SIAM Journal of Numerical Analysis*, Vol. 14, pp. 600–611, 1977.

[281] K. Watanabe and S. G. Tzafestas, "New computationally efficient formula for backward-pass fixed interval smoother and its UD factorization algorithm," *IEE Proceedings*, Vol. 136D, pp. 73–78, 1989.

[282] D. M. Wiberg and L. A. Campbell, "A discrete-time convergent approximation of the optimal recursive parameter estimator," in *Proceedings of the IFAC Identification and System Parameter Identification Symposium*, Vol. 1, pp. 140–144, 1991, International Federation on Automatic Control, Laxenburg, Austria.

[283] W. S. Widnall and P. A. Grundy, *Inertial Navigation System Error Models*, Technical Report TR-03-73, Intermetrics, Cambridge, MA, May 1973.

[284] W. S. Widnall and P. K. Sinha, "Optimizing the gains of the baro-inertial vertical channel," *AIAA Journal of Guidance and Control*, Vol. 3, pp. 172–178, 1980.

[285] N. Wiener, *Time Series*, MIT Press, Cambridge, MA, 1964 (originally published in 1949 as *Extrapolation, Interpolation, and Smoothing of Stationary Time Series with Engineering Applications*).

[286] N. Wiener, *Ex-Prodigy: My Childhood and Youth*, MIT Press, Cambridge, MA, 1964.

[287] N. Wiener, *I Am a Mathematician*, MIT Press, Cambridge, MA, 1964.

[288] A. S. Willsky, *Digital Signal Processing and Control and Estimation Theory: Points of Tangency, Areas of Intersection, and Parallel Directions*, MIT Press, Cambridge, MA, 1979.

[289] M. A. Woodbury, *Inverting Modified Matrices*, Memorandum Report 42, Statistical Research Group, Princeton University, Princeton, NJ, 1950.

[290] Y. Wu, M. Wu, D. Hu and X. Hu, "An improvement to unscented transformation," in *AI 2004: Advances in Artificial Intelligence—17th Australian Joint Conference on Artificial Intelligence,* Cairns, Australia, pp. 1024–1029, Springer Lecture. Notes in Computer Science, Berlin, 2004.

[291] P. C. Young, *Recursive Estimation and Time Series*, Springer-Verlag, New York, 1984.

[292] L. A. Zadeh and C. A. Desoer, *Linear System Theory*, McGraw-Hill, New York, 1963.

[293] L. A. Zadeh and J. R. Ragazzini, "An extension of Wiener's theory of prediction," *Journal of Applied Physics*, Vol. 21, pp. 645–655, 1950.

[294] R. Zurmühl, *Matrizen*, Academic Press, Orlando, FL, 1961.

INDEX

Kalman Filtering: Theory and Practice using MATLAB®, Third Edition. By Mohinder S. Grewal and Angus P. Andrews