KALMAN FILTERING

KALMAN FILTERING

Theory and Practice Using MATLAB®

Fourth Edition

MOHINDER S. GREWAL

ANGUS P. ANDREWS

WILEY

Copyright © 2015 by John Wiley & Sons, Inc. All rights reserved
Published by John Wiley & Sons, Inc., Hoboken, New Jersey
Published simultaneously in Canada

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning, or otherwise, except as permitted under Section 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923, (978) 750-8400, fax (978) 750-4470, or on the web at www.copyright.com. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, (201) 748-6011, fax (201) 748-6008, or online at http://www.wiley.com/go/permission.

Limit of Liability/Disclaimer of Warranty: While the publisher and author have used their best efforts in preparing this book, they make no representations or warranties with respect to the accuracy or completeness of the contents of this book and specifically disclaim any implied warranties of merchantability or fitness for a particular purpose. No warranty may be created or extended by sales representatives or written sales materials. The advice and strategies contained herein may not be suitable for your situation. You should consult with a professional where appropriate. Neither the publisher nor author shall be liable for any loss of profit or any other commercial damages, including but not limited to special, incidental, consequential, or other damages.

For general information on our other products and services or for technical support, please contact our Customer Care Department within the United States at (800) 762-2974, outside the United States at (317) 572-3993 or fax (317) 572-4002.

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print may not be available in electronic formats. For more information about Wiley products, visit our web site at www.wiley.com.

MATLAB® is a trademark of The MathWorks, Inc. and is used with permission. The MathWorks does not warrant the accuracy of the text or exercises in this book. This book's use or discussion of MATLAB® software or related products does not constitute endorsement or sponsorship by The MathWorks of a particular pedagogical approach or particular use of the MATLAB® software.

Library of Congress Cataloging-in-Publication Data:

Grewal, Mohinder S.
Kalman filtering : theory and practice using MATLAB / Mohinder S. Grewal, Angus P. Andrews. – Fourth edition.
pages cm
Includes index.
ISBN 978-1-118-85121-0 (cloth)
1. Kalman filtering. 2. MATLAB. I. Andrews, Angus P. II. Title.
QA402.3.G695 2015
629.8′312–dc23

2014020208

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

CONTENTS

Preface to the Fourth Edition **ix**

Acknowledgements **xiii**

List of Abbreviations **xv**

1 Introduction **1**
1.1 Chapter Focus, 1
1.2 On Kalman Filtering, 1
1.3 On Optimal Estimation Methods, 6
1.4 Common Notation, 28
1.5 Summary, 30
Problems, 31
References, 34

2 Linear Dynamic Systems **37**
2.1 Chapter Focus, 37
2.2 Deterministic Dynamic System Models, 42
2.3 Continuous Linear Systems and their Solutions, 47
2.4 Discrete Linear Systems and their Solutions, 59
2.5 Observability of Linear Dynamic System Models, 61
2.6 Summary, 66
Problems, 69
References, 71

3 Probability and Expectancy 73
3.1 Chapter Focus, 73
3.2 Foundations of Probability Theory, 74
3.3 Expectancy, 79
3.4 Least-Mean-Square Estimate (LMSE), 87
3.5 Transformations of Variates, 93
3.6 The Matrix Trace in Statistics, 102
3.7 Summary, 106
Problems, 107
References, 110

4 Random Processes 111
4.1 Chapter Focus, 111
4.2 Random Variables, Processes, and Sequences, 112
4.3 Statistical Properties, 114
4.4 Linear Random Process Models, 124
4.5 Shaping Filters (SF) and State Augmentation, 131
4.6 Mean and Covariance Propagation, 135
4.7 Relationships Between Model Parameters, 145
4.8 Orthogonality Principle, 153
4.9 Summary, 157
Problems, 159
References, 167

5 Linear Optimal Filters and Predictors 169
5.1 Chapter Focus, 169
5.2 Kalman Filter, 172
5.3 Kalman–Bucy Filter, 197
5.4 Optimal Linear Predictors, 200
5.5 Correlated Noise Sources, 200
5.6 Relationships Between Kalman and Wiener Filters, 201
5.7 Quadratic Loss Functions, 202
5.8 Matrix Riccati Differential Equation, 204
5.9 Matrix Riccati Equation in Discrete Time, 219
5.10 Model Equations for Transformed State Variables, 223
5.11 Sample Applications, 224
5.12 Summary, 228
Problems, 232
References, 235

6 Optimal Smoothers 239
6.1 Chapter Focus, 239
6.2 Fixed-Interval Smoothing, 244
6.3 Fixed-Lag Smoothing, 256
6.4 Fixed-Point Smoothing, 268

6.5 Summary, 275
Problems, 276
References, 278

7 Implementation Methods 281
7.1 Chapter Focus, 281
7.2 Computer Roundoff, 283
7.3 Effects of Roundoff Errors on Kalman Filters, 288
7.4 Factorization Methods for "Square-Root" Filtering, 294
7.5 "Square-Root" and *UD* Filters, 318
7.6 *SigmaRho* Filtering, 330
7.7 Other Implementation Methods, 346
7.8 Summary, 358
Problems, 360
References, 363

8 Nonlinear Approximations 367
8.1 Chapter Focus, 367
8.2 The Affine Kalman Filter, 370
8.3 Linear Approximations of Nonlinear Models, 372
8.4 Sample-and-Propagate Methods, 398
8.5 Unscented Kalman Filters (UKF), 404
8.6 Truly Nonlinear Estimation, 417
8.7 Summary, 419
Problems, 420
References, 423

9 Practical Considerations 427
9.1 Chapter Focus, 427
9.2 Diagnostic Statistics and Heuristics, 428
9.3 Prefiltering and Data Rejection Methods, 457
9.4 Stability of Kalman Filters, 460
9.5 Suboptimal and Reduced-Order Filters, 461
9.6 Schmidt–Kalman Filtering, 471
9.7 Memory, Throughput, and Wordlength Requirements, 478
9.8 Ways to Reduce Computational Requirements, 486
9.9 Error Budgets and Sensitivity Analysis, 491
9.10 Optimizing Measurement Selection Policies, 495
9.11 Summary, 501
Problems, 501
References, 502

10 Applications to Navigation 503
10.1 Chapter Focus, 503
10.2 Navigation Overview, 504

10.3 Global Navigation Satellite Systems (GNSS), 510
10.4 Inertial Navigation Systems (INS), 544
10.5 GNSS/INS Integration, 578
10.6 Summary, 588
Problems, 590
References, 591

Appendix A Software **593**
A.1 Appendix Focus, 593
A.2 Chapter 1 Software, 594
A.3 Chapter 2 Software, 594
A.4 Chapter 3 Software, 595
A.5 Chapter 4 Software, 595
A.6 Chapter 5 Software, 596
A.7 Chapter 6 Software, 596
A.8 Chapter 7 Software, 597
A.9 Chapter 8 Software, 598
A.10 Chapter 9 Software, 599
A.11 Chapter 10 Software, 599
A.12 Other Software Sources, 601
References, 603

Index **605**

PREFACE TO THE FOURTH EDITION

This book is designed to provide our readers a working 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 fourth edition, we have added a new chapter on the attributes of probability distributions of importance in Kalman filtering, added two sections with easier derivations of the Kalman gain, added a section on a new *sigmaRho* filter implementation, updated the treatment of nonlinear approximations to Kalman filtering, expanded coverage of applications in navigation, added many more derivations and implementations for satellite and inertial navigation error models, and included many new examples of sensor integration. For readers who may need more background in matrix mathematics, we have included an Appendix B as a pdf file on the companion Wiley web site at `www.wiley.com/go/kalmanfiltering`.

We have also updated the problem sets and incorporated helpful corrections and suggestions from our readers, reviewers, colleagues, and students for the overall improvement of the textbook.

All software has been provided in MATLAB®, 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. The MATLAB development environment also integrates with the Simulink® simulation environment for code verification on specific applications and code translation

to C for the many applications microprocessors with C compilers. Appendix A has descriptions of the MATLAB software included on the companion Wiley web site. 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 is a 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 for use 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 could also be used for self-instruction or for purposes of review by practicing engineers and scientists who are not intimately familiar with the subject. Chapter 1 provides an informal introduction to the general subject matter by way of its history of development and application. Chapters 2–4 cover the essential background material on linear systems, probability, stochastic processes, and random process modeling. These chapters could be covered in a senior-level course in electrical, computer, and systems engineering.

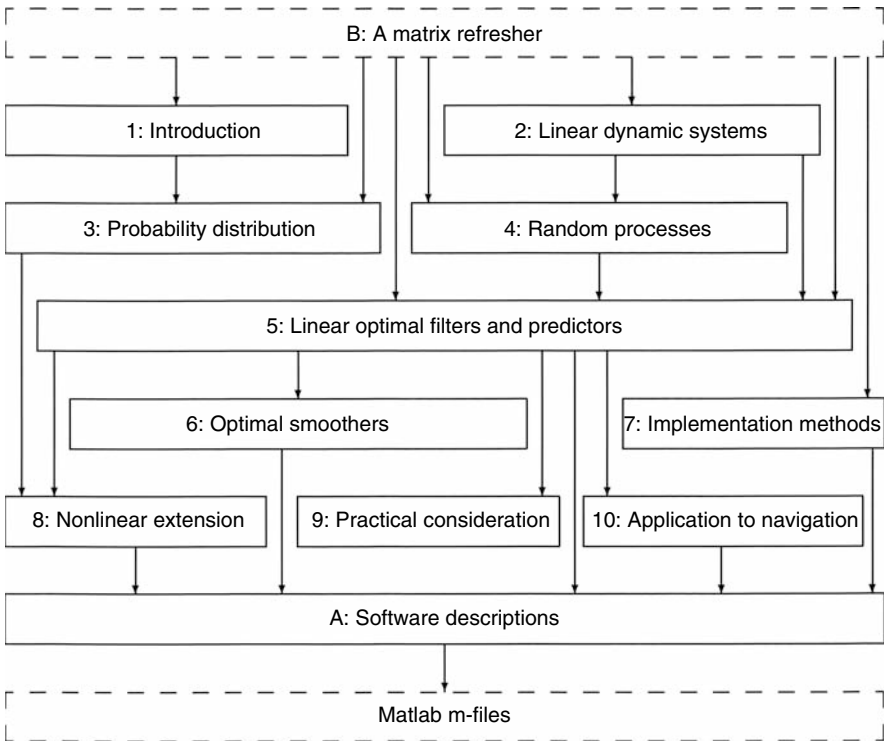
Chapter 5 covers linear optimal filters and predictors, with derivations of the Kalman gain and detailed examples of applications. Chapter 6 is a tutorial-level treatment of optimal smoothing methods based on Kalman filtering models, including more robust implementations. Chapter 7 covers the more recent implementation techniques for maintaining numerical accuracy, with algorithms provided for computer implementation.

Chapter 8 covers approximation methods used for nonlinear applications, including "extended" Kalman filters for "quasilinear" problems and tests for assessing whether extended Kalman filtering is adequate for the proposed application. We also present particle, sigma point, and the "unscented" Kalman filter implementation of Kalman filtering for problems failing the quasilinearity test. Applications of these techniques to the identification of unknown parameters of systems are given as examples. Chapter 9 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 10, we show how to derive and implement different Kalman filtering configurations for Global Navigation Satellite System (GNSS) receivers and inertial navigation systems (INS) and for integrating GNSS receivers with INS.

Chapters 5–9 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.

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. Dashed boxes indicate materials on the Wiley companion web site.



PROF. M. S. GREWAL, PHD, PE
California State University at Fullerton

ANGUS P. ANDREWS, PHD
Senior Scientist (ret.), Rockwell Science Center, Thousand Oaks, California

ACKNOWLEDGEMENTS

The authors express their appreciation to the following individuals for their contributions during the preparation of the core material for this book: 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 thank the following individuals for their review, corrections, and suggestions for improving the second and third editions: Dean Dang, Gordon Inverarity, and Kenneth W. Fertig.

For this fourth edition, we thank Jeffrey Uhlmann and Simon Julier for their assistance on the new material in Chapters 1 and 8, Andrey Podkorytov for his corrections to the Schmidt–Kalman filter, Professor Rudolf E. Kalman for the epigraph to Chapter 1, the late Robert W. Bass (1930–2013) for his corrections to Chapter 1, James Kain for proofreading parts of Chapter 7, John L. Weatherwax for his contributions to the problem set solutions, and Edward H. Martin for providing some early history on GNSS/INS integration.

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

—M. S. G., A. P. A

LIST OF ABBREVIATIONS USED

ANSI, American National Standards Institute

arc-sec, second of arc

BMFLS, Biswas–Mahalanabis fixed-lag smoother

bps, bits per second

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*)

CDMA, code-division multiple access (communications protocol)

dB, decibel

ed., editor or edition

EKF, extended Kalman filter

ENU, east-north-up (coordinates)

f, foot (0.3048 m)

FDMA, frequency-division multiple access (communications protocol)

flops, floating-point operations per second

FLS, fixed-lag smoother

FPS, fixed point smoother

g, 9.80665 m/s^2

gHz, gigahertz

GMLE, Gaussian maximum-likelihood estimator

GNSS, global navigation satellite system

GPS, Global Positioning Service, a GNSS operated by the US Department of Defense

h, hour
Hz, hertz (cycles per second)
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
LGMLE, linear Gaussian maximum-likelihood estimator
LMSE, least-mean-square estimator
LQ, linear quadratic [estimator]
m, meter
MAP, maximum *a posteriori* probability (estimator).
max, maximum
mHz, megahertz
mi, mile
min, minute of time, or minimum
ML, maximum likelihood
MLE, maximum likelihood estimator
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
SLRD, Schweppe likelihood-ration detection
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**,** micrometer (10^{-6} m) or micro (10^{-6} [units])

1

INTRODUCTION

Once you get the physics right, the rest is mathematics.

—Rudolf E. Kalman
Kailath Lecture, Stanford University, May 11, 2009

1.1 CHAPTER FOCUS

This chapter presents a preview of where we are heading, some history of how others got there before us, an overview showing how all the material fits together, and a common notation and nomenclature to make it more apparent.

1.2 ON KALMAN FILTERING

1.2.1 First of All: What Is a Kalman Filter?

Theoretically, it has been called the *linear least mean squares estimator* (LLSME) because it minimizes the mean-squared estimation error for a linear stochastic system using noisy linear sensors. It has also been called the *linear quadratic estimator* (LQE) because it minimizes a quadratic function of estimation error for a linear dynamic system with white measurement and disturbance noise. Even today, more than half a century after its discovery, it remains a unique accomplishment in

Kalman Filtering: Theory and Practice Using MATLAB®, Fourth Edition.
Mohinder S. Grewal and Angus P. Andrews.
© 2015 John Wiley & Sons, Inc. Published 2015 by John Wiley & Sons, Inc.

the history of estimation theory. It is the only practical finite-dimensional solution to the real-time optimal estimation problem for stochastic systems, and it makes very few assumptions about the underlying probability distributions except that they have finite means and second central moments (covariances). Its mathematical model has been found to represent a phenomenal range of important applications involving noisy measurements for estimating the current conditions of dynamic systems with less-than-predictable disturbances. Although many approximation methods have been developed to extend its application to less-than-linear problems, and despite decades of dedicated research directed at generalizing it for nonlinear applications, no comparable general solution[1] for nonlinear problems has been found.

Practically, the Kalman filter is one of the great discoveries of *mathematical engineering*, which uses mathematical modeling to solve engineering problems—in the much same way that mathematical physics is used to solve physics problems, or computational mathematics is used for solving efficiency and accuracy problems in computer implementations.

Its early users would come to consider the Kalman filter to be the greatest discovery in practical estimation theory in the twentieth century, and its reputation has continued to grow over time. As an indication of its ubiquity, a *Google*®; web search for "Kalman filter" or "Kalman filtering" produces more than a million hits. One reason for this is that the Kalman filter has enabled human kind 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 monitoring and 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 that you want to control, and the Kalman filter provides the mathematical framework for inferring the unmeasured variables 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 flood, the trajectories of celestial bodies, or the prices of traded commodities and securities. It has become a universal tool for integrating different sensor and/or data collection systems into an overall optimal solution.

As an added bonus, the Kalman filter model can be used as a tool for assessing the relative accuracy of alternative sensor system designs for likely scenarios of dynamic system trajectories. Without this capability, development of many complex sensor systems (including Global Navigation Satellite Systems) may not have been possible.

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

1. *It is only a tool*. It does not solve any problem all by itself, although it can make it easier for you to do it. 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

[1]However, a somewhat limited finite-dimensional nonlinear solution has been found [1].

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 that you can apply it correctly and efficiently.

2. *It is a computer program*. It has been called "ideally suited to digital computer implementation" [2], 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 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.
3. *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 mean-squared uncertainties arising from random dynamic perturbations and sensor noise. 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 were seeking, then there is no need for you to read the rest of the book. If you need to understand Kalman filters well enough to use them effectively, then please read on!

1.2.2 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 and was used to denote the material that was used as a filter for liquids.) Originally, a filter solved the problem of separating unwanted components of 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. 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 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.

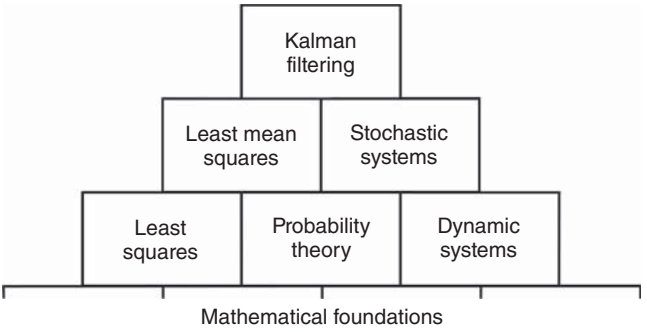


Figure 1.1 Foundational concepts in Kalman filtering.

1.2.3 Its Mathematical Foundations

Figure 1.1 depicts the essential subjects forming the foundations for Kalman filtering theory. Although this 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[2] (matrix theory, in Appendix B on the Wiley web site).

1.2.4 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 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 expressing 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]It is best that one not examine the bottommost 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[a] receiver

[a]*Abbreviation*: GNSS, Global Navigation Satellite System.

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 performance 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 these parameters may be used in assessing its performance as a function of the "design parameters" of an estimation system, such as

1. the types of sensors to be used,
2. the locations and orientations of the various sensor types with respect to the system to be estimated,
3. the allowable noise characteristics of the sensors,
4. the prefiltering methods for smoothing sensor noise,
5. the data sampling rates for the various sensor types, and
6. the level of model simplification to reduce implementation requirements.

The analytical capability of the 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.

1.3 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 [8, 30], Lainiotis [3], Mendel and Gieseking [4], and Sorenson [55, 56] and the personal accounts of Battin [5] and Schmidt [6]. More recent contributions from the last five discoverers on this list are discussed in Chapters 7 and 8.

1.3.1 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 (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

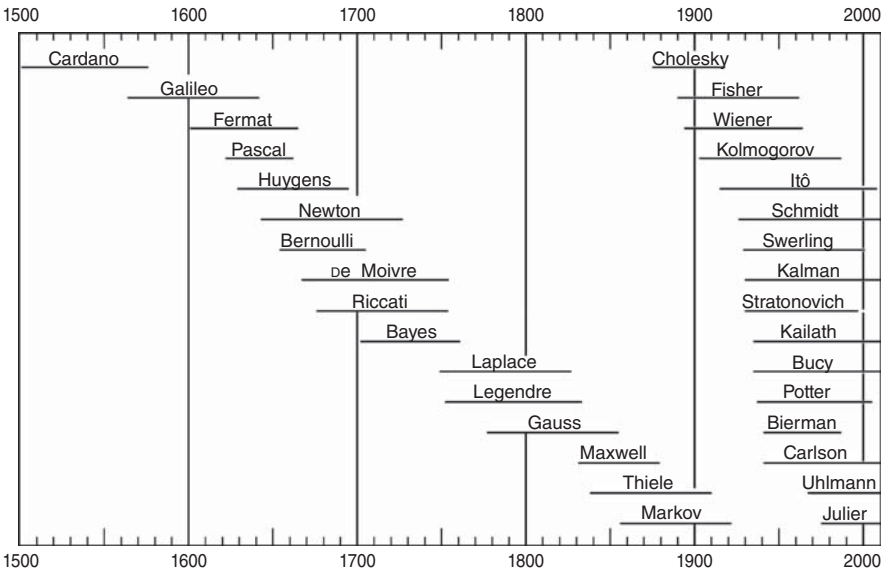


Figure 1.2 Lifelines of some important contributors to estimation technology.

account was put together from several sources, including the account by Baker and Makemson [7].

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 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 there being 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 in 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 Góttingen 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.[3] 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 it in refining his estimates of the orbit of Ceres.

Although Ceres played a significant role in the history of discovery and it 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.

1.3.2 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

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

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:

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

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

[3]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 [8]. 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.

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 are 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

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

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.

1.3.2.1 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 $\mathcal{G} = H^{\mathrm{T}}H$ in this equation is called the *Gramian matrix*.[4] 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 determined uniquely. If its determinant is nonzero, then the solution $\hat{x}$ is uniquely determined.

Example 1.2 (The Gramians of Guier and Weiffenbach) Development of satellite navigation started just after the world's first artificial satellite, Sputnik I, was launched from the Soviet Union on Friday, October 4, 1957. On the following Monday, William Guier (1926–2011) and George Weiffenbach (1921–2003), two scientists at the Applied Physics Laboratory (APL) of Johns Hopkins University, started recording and analyzing the 20 MHz carrier signals from Sputnik I. These signals exhibited noticeable patterns of Doppler shift as the satellite passed from horizon to horizon. Weiffenbach was able to use a spectrum analyzer to track the Doppler frequency shift as the satellite passed from horizon to horizon, generally within a period of several minutes. Curious to understand how the satellite orbit influenced the observed patterns of Doppler shift, Guier and Weiffenbach calculated partial derivatives of Doppler shift with respect to orbital parameters.

For any parameter p_k of the satellite orbit, Guier and Weiffenbach could obtain numerical partial derivatives of the measurable Doppler frequency shift $f_{\mathrm{Dop}}(t)$ at the known receiver location to that parameter be generating an orbit with perturbed value $p_k + \delta_{p,k}$ and calculating the resulting perturbations $\delta_{f,k}(t_i)$ in the Doppler shifts at the receiver at sample times t_i during a satellite pass by, as

$$\frac{\partial f_{\mathrm{Dop}}(t_i)}{\partial p_k} \approx \frac{\delta_{f,k}(t_i)}{\delta_{p,k}}.$$

Small variations $\Delta_{p,k}$ in the orbit parameters should then be approximately related to observable deviations $\Delta_{Dop}(t_i)$ of Doppler shift during one satellite pass by the

[4]Named after 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 probability 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.

linear system of equations

$$\begin{bmatrix} \Delta_{\text{Dop}}\left(t_1\right) \\ \Delta_{\text{Dop}}(t_2) \\ \Delta_{\text{Dop}}(t_3) \\ \vdots \\ \Delta_{\text{Dop}}(t_N) \end{bmatrix} = \underbrace{\begin{bmatrix} \frac{\partial f_{\text{Dop}}(t_1)}{\partial p_1} & \frac{\partial f_{\text{Dop}}(t_1)}{\partial p_2} & \frac{\partial f_{\text{Dop}}(t_1)}{\partial p_3} & \cdots & \frac{\partial f_{\text{Dop}}(t_1)}{\partial p_n} \\ \frac{\partial f_{\text{Dop}}(t_2)}{\partial p_2} & \frac{\partial f_{\text{Dop}}(t_2)}{\partial p_2} & \frac{\partial f_{\text{Dop}}(t_2)}{\partial p_3} & \cdots & \frac{\partial f_{\text{Dop}}(t_2)}{\partial p_n} \\ \frac{\partial f_{\text{Dop}}(t_3)}{\partial p_1} & \frac{\partial f_{\text{Dop}}(t_3)}{\partial p_2} & \frac{\partial f_{\text{Dop}}(t_3)}{\partial p_3} & \cdots & \frac{\partial f_{\text{Dop}}(t_3)}{\partial p_n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_{\text{Dop}}(t_N)}{\partial p_1} & \frac{\partial f_{\text{Dop}}(t_N)}{\partial p_2} & \frac{\partial f_{\text{Dop}}(t_N)}{\partial p_3} & \cdots & \frac{\partial f_{\text{Dop}}(t_N)}{\partial p_n} \end{bmatrix}}_{H} \begin{bmatrix} \Delta_{p,1} \\ \Delta_{p,2} \\ \Delta_{p,3} \\ \vdots \\ \Delta_{p,n} \end{bmatrix},$$

where the matrix H is $N \times n$, N is the number of Doppler frequency shifts observed during one pass, and n is the number of satellite orbit parameters.

The problem of estimating the n unknown orbital parameters $p_1, p_2, p_3, \ldots, p_n$, given the N observations $\Delta_{\text{Dop}}(t_i)$ is similar to the problem faced by Gauss in 1801 in solving for the "Keplerian" orbital parameters of Ceres, in that the available observations span only a small fraction of a complete orbit. Guier and Weiffenbach then did what Gauss had done: they tried Gauss's method of least squares to see whether a suitable estimate could be obtained. That would depend on whether the associated $n \times n$ Gramian matrix

$$G = H^T H$$

is invertible. Gauss had taken several months to obtain his solution, but Guier and Weiffenbach had something Gauss did not have: the use of a Univac 1103A computer.[5]

At first, following the approach of Gauss, the partial derivatives were with respect to the six Keplerian parameters of the Sputnik I orbit. However, effects of gravitational anomalies on satellite orbits were found to be more significant than anticipated, and partial derivatives with respect to the dominant gravitational anomalies were then added to the linearized model. Additional partial derivatives with respect to ionospheric propagation effects and satellite transmitter frequency were also added. In all cases, it could be shown that the associated Gramian matrices G are nonsingular; indicating that the satellite orbit is determinable from the Doppler shift pattern from a single satellite pass by of a receiver with known location. As an added bonus, the solution also provided estimates of anomalies in the gravitational field at satellite altitudes.

In March of 1958, these results were reviewed by Frank McClure (1916–1973), director of APL's Research Center. McClure asked whether this relationship could be inverted to determine the horizontal receiver location, given the Doppler shift history and the satellite ephemeris (orbit description), and Guier and Weiffenbach were able

[5] A vacuum-tube computer with roughly the same capabilities as the IBM 704. It could have one to four banks of 18 kB magnetic core random-access memories in addition to magnetic drum memory, a 36-bit data word, and multiply times in the order of a few tenths of a millisecond.

to show that the 2×2 Gramian matrix for this problem is also nonsingular. That is, given the Doppler frequency shift pattern from one pass by a satellite with known orbit, one can obtain a least-squares solution for the longitude and latitude of the receiver antenna.

This discovery would result in the development of the world's first satellite navigation system: the US Navy's Transit Navigation System. In December of 1956, the US Navy had committed to develop a new class of nuclear powered ballistic missile submarines to be launched in the 1960s, but these submarines would need an accurate position fix before launching their missiles. The Transit Navigation System would fulfill that need. It became operational in the 1960s and remained in operation until it was eclipsed by Global Positioning System (GPS) in the 1990s.

1.3.2.2 Least-Squares Solution In the case that 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 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.

1.3.2.3 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 1.3 (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

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

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

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

which has exactly the same array structure with respect to x as the algebraic least-squares problem. The least-squares solution for x can be found, as before, by taking the derivatives of $\|\varepsilon\|^2$ with respect to the components of 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

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

is nonsingular.

1.3.2.4 Gramian Matrices 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 ($\mathcal{G}$), 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 that 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.

1.3.3 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[6] (1501–1576) performed an accurate analysis of probabilities 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 Nikolayevich 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 theory was the first formal theory of optimal estimation for systems involving random processes.

[6]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*, *Cardan shaft*, or *universal joint*.

1.3.4 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 the Tufts University. He then entered graduate school at the 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 at the 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.[7]

1.3.4.1 Wiener–Kolmogorov Filter Development In the early years of the 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 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.

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

1.3.5 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 the Youngstown College there for 3 years before entering the MIT.

Kalman received his bachelor's and master's degrees in electrical engineering at the 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 [9]. 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 the Columbia University. At that time, Columbia was well known for the work in control theory by John R. Ragazzini, Lotfi A. Zadeh,[8] 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 mathematician Solomon Lefschetz (1884–1972) from Princeton. Lefschetz had been a classmate with rocket pioneer Robert H. Goddard (1882–1945) at the 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 postdoc under Lefschetz at Princeton before coming to the RIAS in 1956. Kalman recommended Richard S. Bucy, who would join him at the RIAS.

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

In late November of 1958, not long after coming to the RIAS, Kalman was returning by train to Baltimore from a visit to Princeton. At around 11 PM, 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*[9] *to the Wiener–Kolmogorov filtering*

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

[9] 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. [10] 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. The rest is history.

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

1. Wiener–Kolmogorov models use the PSD in the frequency domain to characterize the dynamic and statistical properties of a dynamic process. Optimal Wiener–Kolmogorov estimators are derivable from the PSD, which can be estimated from measured system outputs. This assumes 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 Robert W. Bass (1930–2013) [11], who was at the 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 [12] and Ruslan Leont'evich Stratonovich (1930–1997) in 1960 [35].

1.3.5.2 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."[10] 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.

1.3.5.3 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 [13]. Schmidt had known 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* (*EKF*),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 the early part of 1961, Schmidt described his results to Richard H. Battin from the MIT Instrumentation Laboratory (later renamed the Charles Stark Draper Laboratory, then shortened to Draper Laboratory). Battin was already using state-space methods for the design and implementation of astronautical guidance systems, and he made the Kalman filter as part of the Apollo onboard guidance, 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.

1.3.5.4 Other Accomplishments of 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.

[10]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 the UCLA and sponsored jointly by the UCLA and the National Aeronautics and Space Administration (NASA) Dryden Laboratory.

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.

For his many contributions to mathematical engineering, Kalman was awarded the IEEE Medal of Honor in 1974, the IEEE Centennial Medal in 1984, the Steele Prize of the American Mathematical Society in 1987, and the Bellman Prize of the American Automatic Control Council in 1997.

In 1985, the first year the Inamori Foundation awarded its Kyoto Prizes, Kalman was awarded the Kyoto Prize in Advanced Technology. On his visit to Japan to accept the Kyoto 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.

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

Kalman is a member of the US National Academy of Sciences, the US National Academy of Engineering, the American Academy of Arts and Sciences, and a foreign member of the French, Hungarian, and Russian Academies of Sciences.

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 [14].

On February 19, 2008, the US National Academy of Engineering awarded Kalman the Draper Prize, the Nation's most prestigious award in engineering, at an evening ceremony in Washington, DC.

In a ceremony at the White House on October 7, 2009, Kalman was awarded the National Medal of Science by US President Barak Obama.

1.3.5.5 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.

1.3.5.6 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.
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 when 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.

1.3.6 Implementation Methods

1.3.6.1 Numerical Stability Problems The great success of Kalman filtering was not without its problems, not the least of which was 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.

1.3.6.2 Early ad hoc Fixes It was discovered early on[11] 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 [15]. 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. This approach was too easily abused for covering up modeling errors, however.

[11] These fixes were apparently discovered independently by several people. Schmidt [13] and his colleagues at NASA had discovered the use of forced symmetry and "pseudonoise" to counter roundoff effects and credit R. C. K. Lee at Honeywell with the independent discovery of the symmetry effect.

A symmetrized form of the discrete-time Riccati equation was developed by Joseph [16] and used by R. C. K. Lee at Honeywell in 1964. This "structural" reformulation of the Kalman filter equations improved robustness against roundoff errors in some applications, although later methods have performed better on some problems [17].

1.3.6.3 James E. Potter (1937–2005) and Square-Root Filtering The first big breakthrough for 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 7000-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 as

$$P = GG^{\mathrm{T}} \tag{1.14}$$

and expressing the observational update equations in terms of G, rather than P. The result was better numerical stability of the filter implementation. An even more efficient implementation—in terms of *triangular* factors—was published by Bennet in 1967 [18], and the solution was generalized to vector-valued measurements by Andrews in 1968 [19].

Cholesky Factors André-Louis Cholesky[12] (1875–1918) derived an algorithm for solving least-squares problems that included factoring a symmetric positive-definite matrix P as the symmetric product of a triangular matrix C with positive diagonal elements and its transpose:

$$P = CC^{\mathrm{T}}, \tag{1.15}$$

called the *Cholesky decomposition* of P. The triangular factor C is called a Cholesky factor of P.

Generalized Cholesky Factors By convention, only triangular matrices with positive diagonal elements are considered to be Cholesky factors. Otherwise, the solution of Equation 1.14 is not unique. If C is the Cholesky factor of P and M is any orthogonal matrix (so that $MM^{\mathrm{T}} = I$), the matrix

$$G = CM \tag{1.16}$$

[12]Because Cholesky was French, his last name should perhaps be pronounced something like "show-less-KEY," with the accent on the last syllable. Cholesky was a French artillery officer killed in action in World War I, and his algorithm was published posthumously by fellow officer Commandant Benoit [20]. Cholesky may not have been the first to derive the factoring algorithm, but his name was soon attached to it as a matter of respect.

also satisfies the equation

$$GG^{\mathrm{T}} = (CM)(CM)^{\mathrm{T}} \tag{1.17}$$

$$= CMM^{\mathrm{T}}C^{\mathrm{T}} \tag{1.18}$$

$$= CIC^{\mathrm{T}} \tag{1.19}$$

$$= CC^{\mathrm{T}} \tag{1.20}$$

$$= P. \tag{1.21}$$

But, because G is not necessarily triangular with positive diagonal elements, we will call any solution G of $GG^{\mathrm{T}} = P$ a *generalized Cholesky factor* of P.

Matrix Square Roots A square root S of a matrix P satisfies the equation $P = SS$ (i.e., without the transpose on the second factor).

Square-Root Filtering Potter's derivation used a special type of symmetric matrix called an *elementary matrix*, a concept introduced by Householder [21]. Potter factored an elementary matrix as the square of another elementary matrix. In this case, the factors were truly square roots of the factored matrix.

The application on which Potter was working on was for dynamics in space, where there is no appreciable dynamic disturbance noise. In that case, the propagation over a discrete time interval of the covariance matrix P of navigation could be implemented as the double matrix product $\Phi P \Phi^{\mathrm{T}}$, where Φ is a known *state transition matrix* for trajectories in space. Potter could then propagate his generalized Cholesky factor G of P forward in time with a single matrix multiply as ΦG. In doing so, G would no longer remain either a square root or a Cholesky factor of P (unless it remained symmetric). However, this "square-root" appellation has stuck with extensions of Potter's approach, even though the factors involved are generalized Cholesky factors, not matrix square roots.

1.3.6.4 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 [22] on temporal update methods for Cholesky factors. Extensions of square-root covariance and information filters were introduced in Kaminski's 1971 thesis [23] at Stanford University. The first of the triangular factoring algorithms for the observational update was due to Agee and Turner [24], 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 [25], followed by the "square-root-free" algorithm of Bierman in 1974 [26] and the associated temporal update method introduced by Thornton [27]. The computational complexity of the square-root filter for time-invariant systems was greatly simplified by Morf and Kailath [28] soon after that. Specialized

parallel processing architectures for fast solution of the square-root filter equations were developed by Jover and Kailath [29] and others over the next decade, and much simpler derivations of these and earlier square-root implementations were discovered by Kailath [30].

1.3.6.5 Matrix Decomposition, Factorization, and Triangularization These terms are bandied about in square-root filtering, often interchangeably. There are some distinctions, however.

Matrix Decomposition The term *decomposition* is perhaps the most broad. It generally refers to decomposing a matrix into a representation composed of different parts with some useful properties. For example, the "singular value decomposition" (SVD) of a symmetric positive-definite $n \times n$ matrix P yields the product decomposition of P as $P = EDE^{\mathrm{T}}$, where the column vectors of the orthogonal matrix E are the eigenvectors of P and the diagonal matrix D has the corresponding eigenvalues on its diagonal, leading to the alternative representation of $P = EDE^{\mathrm{T}}$ as the "eigenvalue-eigenvector decomposition" of P:

$$P = \sum_{i=1}^{n} \lambda_i e_i e_i^{\mathrm{T}},$$

where the λ_i are the (positive) eigenvalues of P and the e_i are the associated eigenvectors.[13] The SVD, like many other factorization methods used in "square-root" filtering, is also used for solving least-squares problems [31]. The so-called "QR decomposition" of a matrix is another used for solving least-squares problems. It factors a matrix as the product of an orthogonal matrix (Q) and a 'triangular"[14] matrix R (i.e., with zeros either above or below the main diagonal). (However, this notation does conflict with standard notation for Kalman filtering.) The Cholesky decomposition also produces triangular factors, but the term *decomposition* by itself does not imply matrix factoring. For any square matrix S, for example, the symmetric–antisymmetric decomposition

$$S = \underbrace{\frac{1}{2}(S + S^{\mathrm{T}})}_{\text{sym.}} + \underbrace{\frac{1}{2}(S - S^{\mathrm{T}})}_{\text{anti.}}$$

decomposes S as the sum of its symmetric and antisymmetric parts.

Matrix Factorization *Factorization* is a term used by Gerald Bierman (1941–1987) for methods to factor a matrix into a product of matrices with more useful properties for Kalman filtering implementation [32]. For example, the so-called "UD

[13] This eigenvalue-eigenvector decomposition is a property of all "normal" matrices, defined as the square matrices S such that $SS^{\mathrm{T}} = S^{\mathrm{T}}S$.

[14] See Chapter 7 and Appendix B (on the Wiley web site) for further discussions of triangular forms.

decomposition" used by Bierman factors a symmetric positive-definite matrix P as

$$P = UDU^{\mathrm{T}},$$

where D is diagonal with positive diagonal entries and U is the "unit triangular matrix" (i.e., triangular with ones along its main diagonal). "Factorization" generally refers to the algorithmic methods used for obtaining the result, often (but not always) done "in-place" (i.e., in memory, overwriting the input matrix with the factor(s)). For example, UD factorization can overwrite the diagonal of the input matrix with D and off-diagonal terms with those of U (because the diagonal of U is known to contain only ones).

Matrix Triangularization The term *triangularization* refers to factorization in which the resulting factor is triangular. It is used for "QR decompositions" performed in-place, destroying the original matrix and replacing it with its triangular factor (R). The orthogonal transformation Q is not saved, but the operations used to render the effect of Q tend to be well conditioned numerically. The sequence of operations performed in-place is called *triangularization* of the original matrix. Triangularization methods derived by Givens [33], Householder [21], and Gentleman [33] are used to make Kalman filtering implementations more robust against roundoff errors.

The more useful factorization and triangularization methods for Kalman filtering are described in Chapter 7.

1.3.6.6 Generalizations Linear estimation theory has been extended to nonquadratic error criteria, as well. Optimization with respect to the "sup norm" or H_∞ norm minimizes the maximum error, which is advantageous for applications in which the associated risk is decidedly nonquadratic. The first major application of Kalman filtering (for Apollo navigation to the moon and back) had very hard constraints on atmospheric entry on the return to Earth. Large excursions of the entry angle could result in spacecraft burn-up (too steep) or skip-out (too shallow). An H_∞ estimator might have been more appropriate under those circumstances, but it had not been developed yet.

For a more expansive view of linear estimation methods, see Kailath et al. [34] and the references therein.

1.3.7 Nonlinear Approximations

It is human nature to use successful approaches to problem solving within a limited context on problems outside that context. The Kalman filter is no exception to this rule. 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 and mathematically correct approach comparable to the Kalman filter. Although it was

originally derived for linear problems, the Kalman filter is habitually applied to nonlinear problems by using various approximation methods. 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 8.

1.3.7.1 Extended Kalman Filtering (EKF) for Quasilinear Problems EKF 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 being *quasilinear* and sufficiently dominated by linearity within the expected range of variation that unmodeled errors due to linear approximation are insignificant compared to the modeled errors due to dynamic uncertainty and sensor noise. Methods for verifying whether a problem is sufficiently quasilinear are presented in Chapter 8.

In EKF, 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 [13] 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 8.

1.3.7.2 Higher Order Approximations Approaches using higher order expansions of the filter equations (i.e., beyond the linear terms) have been derived by Stratonovich [35], Kushner [36], Bucy [37], Bass et al. [38], and others for quadratic nonlinearities, and by Wiberg and Campbell [39] for terms through third order. However, none of these has proven to be very practical.

1.3.7.3 Sampling-Based Methods for Nonlinear Estimation The Kalman filtering methodology has been further extended to problems for which EKF exhibits unacceptable errors. The general approach to approximating nonlinear propagation of the Riccati equation solution is by using representative samples of state variables—as opposed to linearized propagation of the mean (i.e., the estimated state) and covariance matrix of the distribution.

In the 1940s, mathematician Stanislaw Ulam conceived the idea of using pseudorandom sampling to characterize the evolution of neutron distributions in thermonuclear devices. Colleague Nicholas Metropolis coined the term *Monte Carlo*[15] for such methods. Much of the initial development of Monte Carlo techniques occurred at the Los Alamos Laboratory, where adequate computer resources were then becoming

[15]The name refers to the Monaco Monte Carlo gambling casino, which uses pseudorandom methods to transform the distribution of wealth among its players.

available. Others involved in this development at Los Alamos included 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*, the samples can be based on the eigenvectors and eigenvalues (usually represented by the symbol σ^2) of the covariance matrix.
3. *Unscented transform methods* select samples using the Cholesky decomposition of the covariance matrix. The resulting filter implementation is called *unscented Kalman filtering,* a terminology introduced by Jeffrey Uhlmann. This approach also includes a nonlinear approximation for the cross-covariance of the predicted state vector and the predicted measurement and weighting parameters that can be adjusted for "tuning" the filter to the particular nonlinearities of the application. Unscented transformations for n-dimensional distributions may use $n + 1$ or $2n + 1$ 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.

The more successful of these methods are described in Chapter 8. The unscented Kalman filter, in particular, has been shown to be efficient and effective for some of the more nonlinear applications—including system identification (i.e., estimation of dynamic model parameters), a notoriously nonlinear and difficult problem.

1.3.8 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 [40], Fokker [41], Planck [42], Kolmogorov [43], Stratonovich [35], Baras and Mirelli [44], 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 [36], Bucy [37], and others using the *stochastic calculus* of Stratonovich or Itô. The theoretical basis for stochastic differential equations was long been hampered by the fact that white noise is not a Riemann-integrable function, but the non-Riemannian *stochastic integrals* of Stratonovich or Itô fixed that.

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.

For a concise but readable treatment of the stochastic calculus for Kalman filtering, see Jazwinski [45].

1.3.9 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 in solving 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 type of random process for modeling events or objects that are distributed over time and/or space, such as the arrivals[16] 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 [46]. 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 Gaussian-like closure assumption about the relationships between moments. 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.

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

1.4 COMMON 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. It is sometimes confusing but unavoidable.

1.4.1 "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.

1.4.2 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 [50]. 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.

1.4.2.1 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

TABLE 1.2 Common Symbols Used in Kalman Filtering

Sources*			
(a)	(b)	(c)	Symbol Definition
F	F	A	Dynamic coefficient matrix of continuous linear differential equation defining dynamic system
G	I	B	Coupling matrix between random process noise and state of linear dynamic system
H	M	C	Measurement sensitivity matrix defining the linear relationship between 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

* (a) This book and References 2, 47, 48 and 49.
(b) Reference 50.
(c) References 51–53, and 54.

TABLE 1.3 Special State-Space Notation

This Book	Other Sources	Definition of Notational Usage
x	$\underline{x}$, $\vec{x}$, $\mathbf{x}$	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
$\hat{x}$	$\mathrm{E}\langle x\rangle$, $\overline{x}$	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)

lists the notation used in this book (first 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[17] 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 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.

1.4.3 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. [2] 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.

TABLE 1.4 Common Notation for Array Dimensions

Symbol	Vector Name	Dimensions	Symbol	Matrix Name	Dimensions	
					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

[17] This use of 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 it would become, they might have named them otherwise.

1.5 SUMMARY

The *Kalman filter* is an estimator used to estimate the *state* of a *linear dynamic system* perturbed by *white noise* using measurements that are *linear* functions of the system state but corrupted by *additive* 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, discovered by Gauss (and others) around the end of the eighteenth century. 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 power spectral densities 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 and particle filtering can be used to extend Kalman filtering beyond the *quasilinear* estimation problems that are solvable by *extended Kalman filtering.*

The *Unscented Kalman Filter* has about the same computational complexity as the extended Kalman filter and essentially the same numerical stability as square-root filtering but with potentially greater robustness against nonlinear effects.

PROBLEMS

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

$$2 = a + b$$
$$4 = 3a + b$$
$$1 = 2a + b.$$

(a) Express the system of equations 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

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

for inverting symmetric 2×2 matrices.]

(d) Take the matrix product

$$A^{\mathrm{T}} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$$

for the A derived in (a).

(e) Calculate the least-squares solution

$$\begin{bmatrix} \hat{a} \\ \hat{b} \end{bmatrix} = [A^{\mathrm{T}}A]^{-1}A^{\mathrm{T}} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$$

for the $[A^{\mathrm{T}}A]^{-1}$ derived in (c) and

$$A^{\mathrm{T}} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$$

derived in (d).

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

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

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

$$\begin{aligned} z_1 &= a \times 1\Delta t + b \\ z_2 &= a \times 2\Delta t + b \\ z_3 &= a \times 3\Delta t + b \\ &\vdots \\ z_N &= a \times N\Delta t + b, \end{aligned}$$

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) Derive a general 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.

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:

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

See if you can help him on 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 $1 \le j \le n$, given the least integrated squared approximation error

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

You may assume the equalities

$$\int_0^{2\pi} d\theta = 2\pi$$
$$\int_0^{2\pi} \cos(j\theta)\cos(k\theta)\, d\theta = \begin{cases} 0, & j \ne k \\ \pi, & j = k, \end{cases}$$
$$\int_0^{2\pi} \sin(j\theta)\sin(k\theta)\, d\theta = \begin{cases} 0, & j \ne k \\ \pi, & j = k, \end{cases}$$
$$\int_0^{2\pi} \cos(j\theta)\sin(k\theta)\, d\theta = 0, 0 \le j \le n,\ 1 \le k \le n,$$

as given.

REFERENCES

[1] V. Beneš and R. J. Elliott, "Finite-dimensional solutions of a modified Zakai equation," *Mathematics of Control, Signals, and Systems*, Vol. 9, pp. 341–351, 1996.

[2] 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.

[3] D. G. Lainiotis, "Estimation: a brief survey," *Information Sciences*, Vol. 7, pp. 191–202, 1974.

[4] 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.

[5] R. H. Battin, "Space guidance evolution—a personal narrative," *AIAA Journal of Guidance and Control*, Vol. 5, pp. 97–110, 1982.

[6] S. F. Schmidt, "The Kalman Filter: its recognition and development for aerospace applications," *AIAA Journal of Guidance and Control*, Vol. 4, No. 1, pp. 4–7, 1981.

[7] R. M. L. Baker and M. W. Makemson, *An Introduction to Astrodynamics*, Academic Press, New York, 1960.

[8] 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.

[9] R. E. Kalman, *Phase-Plane Analysis of Nonlinear Sampled-Data Servomechanisms*, M.S. thesis, Department of Electrical Engineering, Massachusetts Institute of Technology, Cambridge, MA, 1954.

[10] J. H. Laning Jr. and R. H. Battin, *Random Processes in Automatic Control*, McGraw-Hill, New York, 1956.

[11] R. W. Bass, "Some reminiscences of control and system theory in the period 1955–1960," talk presented at *The Southeastern Symposium on System Theory*, University of Alabama at Huntsville, 18 March 2002.

[12] 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.

[13] 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, Ames Research Center, Moffett Field, California, 1985.

[14] A. C. Antoulas, Ed., *Mathematical System Theory, The Influence of R. E. Kalman*, Springer-Verlag, Berlin, 1991.

[15] 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.

[16] R. S. Bucy and P. D. Joseph, *Filtering for Stochastic Processes, with Applications to Guidance*, John Wiley & Sons, Inc., New York, 1968.

[17] C. L. Thornton and G. J. Bierman, *A Numerical Comparison of Discrete Kalman Filtering Algorithms: An Orbit Determination Case Study*, JPL Technical Memorandum, Pasadena, CA, pp. 33–771, 1976.

[18] J. M. Bennet, "Triangular factors of modified matrices," *Numerische Mathematik*, Vol. 7, pp. 217–221, 1963.

[19] A. Andrews, "A square root formulation of the Kalman covariance equations," *AIAA Journal*, Vol. 6, pp. 1165–1166, 1968.

[20] Commandant Benoit, "Note sur une méthode de résolution des équations normales provenant de l'application de la méthode des moindres carrés à un systéme d'équations linéaires en nombre infèrieur à celui des inconnues," *(Procédé du Commandant Cholesky), Bulletin Géodésique*, Vol. 2, pp. 67–77, 1924.

[21] A. S. Householder, "Unitary triangularization of a nonsymmetric matrix," *Journal of the Association for Computing Machinery*, Vol. 5, pp. 339–342, 1958.

[22] 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.

[23] P. G. Kaminski, *Square Root Filtering and Smoothing for Discrete Processes*, PhD thesis, Stanford University, 1971.

[24] 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 Tech. Rep. No. 38, Oct. 1972.

[25] N. A. Carlson, "Fast triangular formulation of the square root filter," *AIAA Journal*, Vol. 11, No. 9, pp. 1259–1265, 1973.

[26] G. J. Bierman, *Factorization Methods for Discrete Sequential Estimation*, Academic Press, New York, 1977.

[27] C. L. Thornton, *Triangular Covariance Factorizations for Kalman Filtering*, PhD thesis, University of California at Los Angeles, School of Engineering, 1976.

[28] M. Morf and T. Kailath, "Square root algorithms for least squares estimation," *IEEE Transactions on Automatic Control*, Vol. AC-20, pp. 487–497, 1975.

[29] J. M. Jover and T. Kailath, "A parallel architecture for Kalman filter measurement update and parameter update," *Automatica*, Vol. 22, pp. 783–786, 1986.

[30] T. Kailath, "State-space modelling: square root algorithms," in *Systems and Control Encyclopedia* (M. G. Singh, Ed.), Pergamon, Elmsford, NY, 1984.

[31] G. H. Golub and C. Reinsch, "Singular value decomposition and least squares solutions," *Numerische Mathematik*, Vol. 14, No. 5, pp. 403–420, 1970.

[32] 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.

[33] 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.

[34] T. Kailath, A. H. Sayed and B. Hassibi, *Linear Estimation,* Prentice-Hall, Upper Saddle River, NJ, 2000.

[35] R. L. Stratonovich, *Topics in the Theory of Random Noise*, (R. A. Silverman, Ed.), Gordon & Breach, New York, 1963.

[36] H. J. Kushner, "On the differential equations satisfied by conditional probability densities of Markov processes," *SIAM Journal on Control, Series A*, Vol. 2, pp. 106–119, 1964.

[37] R. S. Bucy, "Nonlinear filtering theory," *IEEE Transactions on Automatic Control*, Vol. AC-10, pp. 198–206, 1965.

[38] 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.

[39] D. M. Wiberg and L. A. Campbell, "A discrete-time convergent approximation of the optimal recursive parameter estimator," *Proceedings of the IFAC Identification and System Parameter Identification Symposium*, Vol. 1, pp. 140–144, 1991.

[40] 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.

[41] A. D. Fokker, "Die mittlerer Energie rotierender elektrischer Dipole im Strahlungsfeld," *Annelen der Physik*, Vol. 43, pp. 810–820, 1914.

[42] M. Planck, "Über einen Satz der statistischen Dynamik und seine Erweiterung in der Quantentheorie," *Sitzungsberichte d. König. Preussischen Akademie der Wissenschaft*, Vol. 24, pp. 324–341, 1917.

[43] A. A. Kolmogorov, "Über die analytichen Methoden in der Wahrscheinlichkeitsrechnung," *Mathematische Annelen*, Vol. 104, pp. 415–458, 1931.

[44] J. Baras and V. Mirelli, *Recent Advances in Stochastic Calculus*, Springer-Verlag, New York, 1990.

[45] A. H. Jazwinski, *Stochastic Processes and Filtering Theory*, Academic Press, New York, 1970.

[46] J. M. Richardson and K. A. Marsh, "Point process theory and the surveillance of many objects," in *Proceedings of the 1991 Symposium on Maximum Entropy and Bayesian Methods*, Seattle University, 1991.

[47] B. D. O. Anderson and J. B. Moore, *Optimal Filtering*, Prentice-Hall, Englewood Cliffs, NJ, 1979.

[48] R. G. Brown and P. Y. C. Hwang, *Introduction to Random Signals and Applied Kalman Filtering: With MATLAB Exercises and Solutions*, 4th ed., John Wiley & Sons, Inc., New York, 2007.

[49] D. E. Catlin, *Estimation, Control, and the Discrete Kalman Filter*, Springer-Verlag, New York, 1989.

[50] R. E. Kalman, "A new approach to linear filtering and prediction problems," *ASME Journal of Basic Engineering*, Vol. 82, pp. 34–45, 1960.

[51] A. V. Balakrishnan, *Kalman Filtering Theory*, Optimization Software, New York, 1987.

[52] K. Brammer and G. Siffling, *Kalman-Bucy Filters*, Artech House, Norwood, MA, 1989.

[53] C. K. Chui and G. Chen, *Kalman Filtering with Real-Time Applications*, Springer-Verlag, New York, 1987.

[54] T. Kailath, *Linear Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1980.

[55] H. W. Sorenson, "Least-squares estimation: From Gauss to Kalman," *IEEE Spectrum*, Vol. 7, pp. 63–68, 1970.

[56] H. W. Sorenson, "On the development of practical nonlinear filters," *Information Sciences*, Vol. 7, pp. 253–270, 1974.

2

LINEAR DYNAMIC SYSTEMS

All the effects of nature are only mathematical results of a small number of immutable laws.

—Pierre-Simon, marquis de Laplace (1749–1827)

2.1 CHAPTER FOCUS

This chapter is about the dynamic models used in Kalman filtering, and 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, such as $F = ma$, its rotational companion $T = M\dot{\omega}$, or Newton's laws of gravitation (used in Example 2.1).

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

Kalman Filtering: Theory and Practice Using MATLAB®, Fourth Edition.
Mohinder S. Grewal and Angus P. Andrews.
© 2015 John Wiley & Sons, Inc. Published 2015 by John Wiley & Sons, Inc.

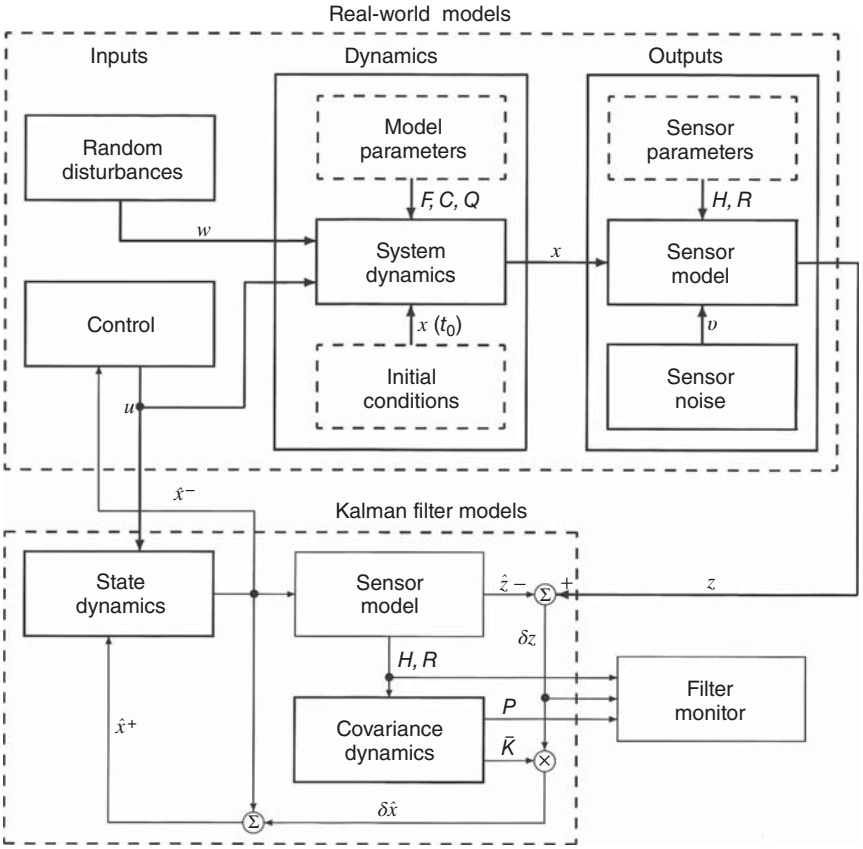


Figure 2.1 Schematic of Kalman filter implementation.

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 the following 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;
  - 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.

TABLE 2.1 Mathematical Models for Dynamic Systems

Model Type	Continuous-Time Model	Discrete-Time Model
Time invariant		
Linear	$\dot{x}(t) = Fx(t) + Cu(t) + w(t)$	$x_k = \Phi x_{k-1} + \Gamma u_{k-1} + w_{k-1}$
	$z(t) = Hx(t) + Du(t) + v(t)$	$z_k = Hx_k + Du_k + v_k$
General	$\dot{x}(t) = f(x(t), u(t)) + w(t)$	$x_k = f(x_{k-1}, u_{k-1}) + w_{k-1}$
	$z(t) = h(x(t), u(t)) + v(t)$	$z_k = h(x_k, u_k) + v_k$
Time varying		
Linear	$\dot{x}(t) = F(t)x(t) + C(t)u(t) + w(t)$	$x_k = \Phi_{k-1}x_{k-1} + \Gamma_{k-1}u_{k-1} + w_{k-1}$
	$z(t) = H(t)x(t) + D(t)u(t) + v(t)$	$z_k = H_k x_k + D_k u_k + v_k$
General	$\dot{x}(t) = f(t, x(t), u(t)) + w(t)$	$x_k = f_k(x_{k-1}, u_{k-1}) + w_{k-1}$
	$z(t) = h(t, x(t), u(t)) + v(t)$	$z_k = h_k(x_k, u_k) + v_k$

System inputs:
u represents known control inputs.
w represents random dynamic disturbances.
v represents random sensor noise.
System outputs:
z represents sensor outputs.

- 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 6.)
  - 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 8.
  - 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 4. 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 4.
- Sensor models are addressed in Chapters 4 and 8 (for nonlinear sensors).
- The Riccati equation is addressed in Chapters 5 and 9, and the equivalent methods for nonlinear applications are covered in Chapter 8.
- Methods for Kalman filter health monitoring (in the box labeled filter monitor) are discussed in Chapter 9.

2.1.2 Models for Dynamic Systems

2.1.2.1 Differential Equations and State Variables Since their introduction by Isaac Newton and Gottfried 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.

2.1.2.2 Other Approaches The whole of dynamic system theory is a subject of considerably more scope than one needs for the present undertaking (Kalman filtering). This chapter will stick to just those concepts that are essential for that purpose, which is the development of the state-space representation for dynamic systems described by the systems of linear differential equations. These are given a somewhat heuristic treatment, without the mathematical rigor often accorded the subject, but

including development and use of the transform methods of functional analysis for solving differential equations 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*[1] 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 4, the inputs are allowed to be nondeterministic (i.e., random), and the focus of the Chapter 5 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 the 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 8). They also include models in continuous time, as defined by differential equations, and discrete time, which is 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. This is how we have learned to apply the laws of physics to represent dynamic systems.

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. Having derived trustworthy models in continuous time, this is how it gets reformulated for Kalman filtering.

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.

[1]The italicized terms in this sentence will be defined more precisely further along.

2.2 DETERMINISTIC 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.

2.2.2 Newtonian Models

The most fundamental state-space models for vehicles or all sorts come from Newtonian mechanics of "rigid" (i.e., infinitely stiff) bodies characterized by their mass and rotational moments of inertia,[2] with any applied forces supplying both accelerations and torques. The distributions of applied forces can usually be decoupled into translational forces and rotational torques, which can be treated independently. Each results in a system of first-order linear differential equations.

2.2.2.1 Rigid-Body Translational Mechanics If x is a position vector for the center of mass of an object, then a state-space model for the dynamics of that point mass has the general form

$$\dot{x} = v \text{ (velocity)} \tag{2.1}$$

$$\dot{v} = a \text{ (acceleration)} \tag{2.2}$$

$$\dot{a} = j_e \text{ (jerk)} \tag{2.3}$$

$$\dot{j}_e = j_o \text{ (jounce)} \tag{2.4}$$

$$\vdots$$

where the series of derivatives can continue as long as the application requires, and the right-hand sides can also represent input accelerations, jerk, and so on. This is where Newton's Laws come in, with $F = ma$, and so on.

[2]Interestingly enough, the Kalman filter also characterizes its error dynamics in terms of their means and moments about their means.

2.2.2.2 Rigid-Body Rotational Mechanics Newtonian rotational mechanics are a bit more complicated because attitude is three dimensional, but its domain folds back on itself, in effect. For example, if you rotate yourself about the vertical axis by 360°, you end up where you started without changing the direction of rotation. Orientation has three independent degrees of freedom (represented by, e.g., roll, pitch, and yaw angles), but its topology is the same as the surface of the unit sphere in four dimensions. It has been quite successfully modeled as such by using four-dimensional quaternians of unit length. Attitude can also be represented by other mathematical objects, including 3×3 orthogonal matrices A, where $A = I$ would represent the reference attitude or origin. The linear dynamic model with A as the orientation variable would then be

$$\dot{A} = A\,\omega \otimes (\text{angular velocity}) \tag{2.5}$$

$$\omega\otimes \overset{\text{def}}{=} \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \tag{2.6}$$

$$\dot{\omega} = \alpha \text{ (angular acceleration)} \tag{2.7}$$

$$\dot{\alpha} = ? \text{ (angular jerk)} \tag{2.8}$$

$$\vdots$$

where, as in the translational case, the series of derivatives can continue as long as the application requires. In this case, the rotational analog of $F = ma$ is $\tau = M\alpha$, where τ is the applied torque and M (the analog of m in translational mechanics) is the positive-definite matrix of body moments of inertia about its center of gravity.

2.2.2.3 Nonrigid Body Dynamic Models Because real-world bodies are not infinitely rigid, they generally have vibration modes with some amount of damping. These vibration modes have their own state-space models, which can be either translational or rotational and generally including coupled vibration modes. These modes can be important dynamic subsystems for some space applications (e.g., space-based telescopes) and for applications with acceleration or rotation sensors attached to the host vehicle frame.[3] There can be similar vibrational issues with fuel sloshing in rockets and aircraft as the tanks empty. In these cases, it is usually necessary to model the dynamics in order to control it.

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

[3]For example, early Atlas missile testing showed that the bending modes of the monocoque frame gave the closed-loop missile attitude control system a pole in the right-half plane.

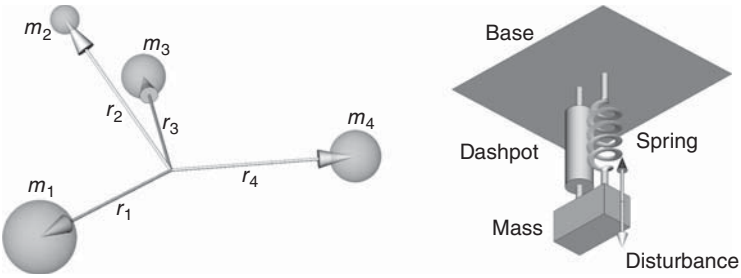


Figure 2.2 Dynamical system examples. (a) Gravitational dynamics and (b) mechanical oscillator.

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 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 (The Harmonic Resonator with Linear Damping) Consider 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.[4] 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

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

[4]Dashpots operate in the regime of "low Reynolds number" hydrodynamics, where damping is linear.

in which time (t) is the differential variable and displacement (δ) is the dependent variable. This equation constrains the dynamical 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 discrete-time linear state dynamic equations derived from linear differential or difference equations.

2.2.3 State Variables and State Equations for Deterministic Systems

The second-order differential equation of the previous example can be transformed to 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

$$\dot{x}(t) = f(t, x(t), u(t)), \tag{2.9}$$

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

$$\begin{aligned}
\dot{x}_1 &= f_1(t, x_1, x_2, x_3, \ \dots\ , x_n, u_1, u_2, u_3, \ \dots\ , u_r),\\
\dot{x}_2 &= f_2(t, x_1, x_2, x_3, \ \dots\ , x_n, u_1, u_2, u_3, \ \dots\ , u_r),\\
\dot{x}_3 &= f_3(t, x_1, x_2, x_3, \ \dots\ , x_n, u_1, u_2, u_3, \ \dots\ , u_r),\\
&\ \ \vdots\\
\dot{x}_n &= f_n(t, x_1, x_2, x_3, \ \dots\ , x_n, u_1, u_2, u_3, \ \dots\ , u_r)
\end{aligned} \tag{2.10}$$

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

2.2.3.1 Homogeneous and Nonhomogeneous Differential Equations The variables u_i in Equations 2.10 can be independent of the state variables x_i, 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_i may become

control inputs that are functions of the x_i (or of the dynamic system outputs, which are functions of the x_i).

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_i) are called *homogeneous.*

2.2.3.2 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 2.10. They are collected into a single n-vector

$$x(t) = \begin{bmatrix} x_1(t) & x_2(t) & x_3(t) & \cdots & x_n(t) \end{bmatrix}^{\mathrm{T}} \tag{2.11}$$

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 [1]. In that sense, the initial value of each state variable represents an independent degree of freedom of the dynamic system. The n values $x_1(t_0), x_2(t_0), x_3(t_0), \ldots, x_n(t_0)$ 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

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

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

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.

2.2.4 Continuous Time and Discrete Time

The dynamic system defined by Equation 2.10 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,

$$x(t_{k+1}) = f(x(t_k), t_k, t_{k+1}), \tag{2.12}$$

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,

$$t_k = t[sub\ 0] + k\Delta t.$$

2.2.4.1 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, as 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.

2.2.5 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. These model represent dynamic systems using three types of variables:

- *Control inputs* u_i, which can be under our control, and therefore known to us.

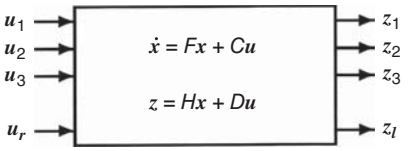


Figure 2.3 Block diagram of a linear dynamic system.

- Internal *state variables* x_i, which were described in the previous section. In most applications, these are "hidden variables," in the sense that they cannot generally be measured directly but must be somehow inferred from what can be measured.
- *Outputs* z_i, 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_i.

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

The common model for a generic linear dynamic systems with inputs and outputs is shown schematically in Figure 2.3.

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

$$\begin{aligned}\dot{x}(t) &= \frac{d}{dt}x(t) \\ &= F(t)x(t) + C(t)u(t),\end{aligned} \tag{2.13}$$

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 2.13 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

$$\frac{d^n y(t)}{dt^n} + f_1(t)\frac{d^{n-1} y(t)}{dt^{n-1}} + \cdots + f_{n-1}(t)\frac{dy(t)}{dt} + f_n(t)y(t) = u(t) \tag{2.14}$$

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 [2], 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

$$x(t) = \left[y(t), \quad \frac{d}{dt}y(t), \quad \frac{d^2}{dt^2}y(t), \quad \ldots, \quad \frac{d^{n-1}}{dt^{n-1}}y(t)\right]^{\mathrm{T}}. \tag{2.15}$$

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

$$\frac{d}{dt}\begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_{n-1}(t) \\ x_n(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ -f_n(t) & -f_{n-1}(t) & -f_{n-2}(t) & \cdots & -f_1(t) \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ \vdots \\ x_n(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} u(t). \tag{2.16}$$

When Equation 2.16 is compared with Equation 2.13, the matrices $F(t)$ and $C(t)$ are easily identified.

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 [3] 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

$$z(t) = H(t)x(t) + D(t)u(t), \tag{2.17}$$

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 2.13 and the output equation 2.17 together form the dynamic equations of the system shown in Table 2.1.

2.3.5 Difference Equations and State-Transition Matrices (STMs)

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

$$\begin{aligned} x(t_{k+1}) &= \phi(t_k, x(t_k), u(t_k)), \\ \phi(t_k, x(t_k), u(t_k)) &= x(t_k) + \psi(t_k, x(t_k), u(t_k)). \end{aligned} \tag{2.18}$$

The second of these (Equation 2.18) has the same general form of the recursive relation shown in Equation 2.12, 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:

$$\begin{aligned} x(t_{k+1}) - x(t_k) &= \Psi(t_k)x(t_k) + C(t_k)u(t_k), \\ x_{k+1} &= \Phi_k x_k + C_k u_k, \\ \Phi_k &= I + \Psi(t_k), \end{aligned} \tag{2.19}$$

where the matrices Ψ and Φ replace the functions ψ and ϕ, respectively. The matrix Φ is called the *state-transition matrix* (*STM*). 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

An STM is a solution of what is called the *homogeneous*[5] matrix equation associated with a given linear dynamic system. Let us first define what homogeneous equations are 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.

[5]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.

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

$$\dot{x}(t) = \frac{d}{dt}[\Phi(t)x(0)] \tag{2.20}$$

$$= \left[\frac{d}{dt}\Phi(t)\right] x(0) \tag{2.21}$$

$$= [F(t)\Phi(t)]x(0) \tag{2.22}$$

$$= F(t)[\Phi(t)x(0)] \tag{2.23}$$

$$= F(t)x(t). \tag{2.24}$$

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 (Toeplitz matrices) The unit upper triangular Toeplitz[6] 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 for the nth-order derivative, $\left(\frac{df}{dt}\right)^n$, the state-space form of which is $\dot{x} = Fx$ with 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 $(d/dt)^n y(t) = 0$.

[6]Named after Otto Toeplitz (1881–1940).

2.3.6.3 Existence and Nonsingularity of Fundamental Solutions If the elements of the matrix $F(t)$ are continuous functions on some interval $0 \leq t \leq T$, then the fundamental solution matrix $\Phi(t)$ is guaranteed to exist and to be nonsingular on an interval $0 \leq t \leq \tau$ 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 ($\Phi(0) = I$) [1].

2.3.6.4 State-Transition Matrices Note that the fundamental solution matrix $\Phi(t)$ transforms any initial state $x(0)$ of the dynamic system to 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

$$\Phi(\tau, t) = \Phi(\tau)\Phi^{-1}(t) \tag{2.25}$$

transforms a solution from time t to the corresponding solution at time τ, as diagrammed in Figure 2.4. Such a matrix is called the *state-transition matrix*[7] for the associated linear homogeneous differential equation. The STM $\Phi(\tau, t)$ 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 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. $(\partial/\partial t)\Phi(\tau, t) = -\Phi(\tau, t)F(t)$.

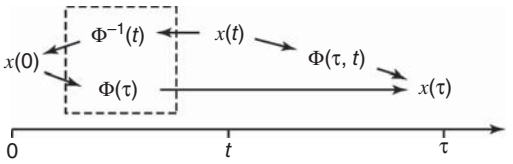


Figure 2.4 The STM as a composition of fundamental solution matrices.

[7]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. An STM is a linear evolution operator.

Example 2.6 (State-Transition Matrix for the Underdamped Harmonic Resonator Model) In 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

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

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

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

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 as

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

The initial values

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

can be solved for a and b as

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

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 2.13 is given by

$$x(t) = \Phi(t, t_0)x(t_0) + \int_{t_0}^{t} \Phi(t, \tau)C(\tau)u(\tau)\, d\tau \tag{2.26}$$

$$= \Phi(t)\Phi^{-1}(t_0)x(t_0) + \Phi(t)\int_{t_0}^{t} \Phi^{-1}(\tau)C(\tau)u(\tau)\, d\tau, \tag{2.27}$$

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 STMs 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 STMs $\Phi(t, \tau)$ will only depend on the differences $t - \tau$. In fact, one can show that

$$\Phi(t, \tau) = e^{F(t-\tau)} \tag{2.28}$$

$$= \sum_{i=0}^{\infty} \frac{(t-\tau)^i}{i!} F^i, \tag{2.29}$$

where $F^0 = I$, by definition. Equation 2.28 uses the matrix exponential function, defined by Equation 2.29.

The solution of the nonhomogeneous equation in this case will be

$$x(t) = e^{F(t-\tau)}x(\tau) + \int_{\tau}^{t} e^{F(t-\sigma)}Cu(\sigma)\, d\sigma \tag{2.30}$$

$$= e^{F(t-\tau)}x(\tau) + e^{Ft}\int_{\tau}^{t} e^{-F\sigma}Cu(\sigma)\, d\sigma. \tag{2.31}$$

2.3.8.2 Using Laplace Transforms Equation 2.29 can be used to derive the STM $\Phi(t)$ using inverse Laplace transforms. The Laplace transform of $\Phi(t) = \exp\ (Ft)$ will be

$$\begin{aligned}
\mathcal{L}\Phi(t) &= \mathcal{L}[\exp\ (Ft)] \\
&= \mathcal{L}\left[\sum_{k=0}^{\infty} \frac{1}{k!} F^k t^k\right] \\
&= \sum_{k=0}^{\infty} \frac{1}{k!} F^k\ [\mathcal{L}t^k] \\
&= \sum_{k=0}^{\infty} \frac{1}{k!} F^k\ \left[\frac{k!}{s^{k+1}}\right] \\
&= \frac{1}{s}\sum_{k=0}^{\infty} (s^{-1}F)^k \\
&= \frac{1}{s}[I - s^{-1}F]^{-1} \\
&= [sI - F]^{-1},
\end{aligned} \tag{2.32}$$

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

$$\Phi(t) = \mathcal{L}^{-1}\{[sI - F]^{-1}\}. \tag{2.33}$$

Equations 2.29 (matrix exponential) and 2.33 (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 this section. The MATLAB® function `expm` uses this method to compute the matrix exponential numerically.
2. Numerical integration of the homogeneous part of the differential equation,

$$\frac{d}{dt}\Phi(t) = F\Phi(t), \tag{2.34}$$

   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,[8] but these are the most important.

Example 2.7 (Laplace Transform Method) We will now demonstrate a Laplace transform method for deriving the STM for nonhomogeneous differential equations and the total solution.

The differential equation model is given as follows for a nonhomogeneous equation:

$$\dot{x}_1(t) = -2x_2(t)$$
$$\dot{x}_2(t) = x_1(t) - 3\ x_2(t) + u(t).$$

The output equation is given as follows:

$$z(t) = x_1(t).$$

These equations can be rewritten in state variable form, as in Equations 2.13 and 2.17.

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

[8]See, for example, Brockett [2], DeRusso et al. [4], Timothy and Bona [5], Evangelisti [6], or Kreindler and Sarachik [7].

$$\dot{x}(t) = \begin{bmatrix} 0 & -2 \\ 1 & -3 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t)$$
$$z(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} x(t) + [0]u(t),$$

where

$$u(t) = \begin{cases} 0, & t < 0, \\ 1, & t \geq 0 \end{cases}$$

is a unit step function at $t = 0$, and the state-vector model coefficient matrices are

$$F = \begin{bmatrix} 0 & -2 \\ 1 & -3 \end{bmatrix}$$
$$C = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$
$$H = \begin{bmatrix} 1 & 0 \end{bmatrix}$$
$$D = [0].$$

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

$$(sI - F) = \begin{bmatrix} s & 2 \\ -1 & s+3 \end{bmatrix}$$
$$(sI - F)^{-1} = \frac{1}{(s^2 + 3s + 2)} \begin{bmatrix} s+3 & -2 \\ 1 & s \end{bmatrix}$$
$$\Phi(t) = \mathcal{L}^{-1}[sI - F]^{-1}$$
$$= \mathcal{L}^{-1} \begin{bmatrix} \dfrac{s+3}{(s+1)(s+2)} & \dfrac{-2}{(s+1)(s+2)} \\ \dfrac{2}{(s+1)(s+2)} & \dfrac{s}{(s+1)(s+2)} \end{bmatrix}$$
$$= \begin{bmatrix} 2e^{-t} - e^{-2t} & 2(e^{-2t} - e^{-t}) \\ e^{-t} - e^{-2t} & 2e^{-2t} - e^{-t} \end{bmatrix}.$$

General Solution Equation 2.30 with $\tau = 0$ becomes

$$x(t) = \Phi(t) \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \int_0^t \Phi(t - \tau) \begin{bmatrix} 0 \\ 1 \end{bmatrix} d\tau$$
$$= \begin{bmatrix} 2e^{-t} - e^{-2t} \\ e^{-t} - e^{-2t} \end{bmatrix} + \int_0^t \begin{bmatrix} 2(e^{-2(t-\tau)}) - e^{-(t-\tau)} \\ 2e^{-2(t-\tau)} - e^{-(t-\tau)} \end{bmatrix} d\tau$$

$$
\begin{aligned}
z(t) &= \begin{bmatrix} 1 & 0 \end{bmatrix} \left\{ \begin{bmatrix} 2e^{-t} - e^{-2t} \\ e^{-t} - e^{-2t} \end{bmatrix} + \int_0^t \begin{bmatrix} 2(e^{-2(t-\tau)} - e^{-(t-\tau)} \\ 2e^{-2(t-\tau)} - e^{-(t-\tau)} \end{bmatrix} d\tau \right\} \\
&= (2e^{-t} - e^{-2t}) + \int_0^t 2[e^{-2(t-\tau)} - e^{-(t-\tau)}] d\tau \\
&= 2e^{-t} - e^{-2t} + 2e^{-t} - e^{-2t} - 1, \ t \geq 0 \\
&= 4e^{-t} - 2e^{-2t} - 1, \ t \geq 0.
\end{aligned}
$$

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 2.34 can be solved numerically by the fourth-order Runge–Kutta method.[9] The ordinary differential equation solvers in MATLAB make use of Runge–Kutta integration methods.

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

$$x(t_k) = \Phi(t_k, t_{k-1})x(t_{k-1}) + \int_{t_{k-1}}^{t_k} \Phi(t_k, \sigma)C(\sigma)u(\sigma) \, d\sigma \tag{2.35}$$

to propagate the state vector between the times of interest.

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

$$x(t_k) = \Phi(t_k, t_{k-1})x(t_{k-1}) + \Gamma(t_{k-1})u(t_{k-1}) \tag{2.36}$$

$$\Gamma(t_{k-1}) = \int_{t_{k-1}}^{t_k} \Phi(t_k, \sigma)C(\sigma) \, d\sigma. \tag{2.37}$$

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

$$
\begin{aligned}
&x_k \stackrel{\text{def}}{=} x(t_k), \quad z_k \stackrel{\text{def}}{=} z(t_k), \quad u_k \stackrel{\text{def}}{=} u(t_k), \quad H_k \stackrel{\text{def}}{=} H(t_k), \\
&D_k \stackrel{\text{def}}{=} D(t_k), \quad \Phi_{k-1} \stackrel{\text{def}}{=} \Phi(t_k, t_{k-1}), \quad G_k \stackrel{\text{def}}{=} G(t_k)
\end{aligned}
$$

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

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

$$x_k = \Phi_{k-1}x_{k-1} + \Gamma_{k-1}u_{k-1}, \tag{2.38}$$

$$z_k = H_k x_k + D_k u_k. \tag{2.39}$$

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 2.38 and 2.39 are independent of the discrete-time index k as well. In that case, the solution can be written in closed form as

$$x_k = \Phi^k x_0 + \sum_{i=0}^{k-1} \Phi^{k-i-1}\Gamma u_i, \tag{2.40}$$

where Φ^k is the kth power of Φ.

Inverse z-transform solution The matrix powers Φ^k in Equation 2.40 can be computed using z-transforms as

$$\Phi^k = \mathcal{Z}^{-1}[(zI - \Phi)^{-1}z^k], \tag{2.41}$$

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 STM ($\Phi(t)$) for linear time-invariant systems in continuous time. The analogous formulas for the inverse transforms are

$$\begin{aligned}\Phi(t) &= \mathcal{L}^{-}1(sI - F)^{-1} \text{ (inverse Laplace transform)}\\ \Phi^k &= \mathcal{Z}^{-1}(zI - F)^{-1}z^k \text{ (inverse z-transform)}\\ &= \sum \text{residuals of } [(zI - F)^{-1}z^k] \text{ (Cauchy's residue theorem)}\end{aligned}$$

Example 2.8 (Solution using z-transforms) Consider Equation 2.38 with

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

The matrices

$$[zI - \Phi]^{-1} = \begin{bmatrix}z & -1\\6 & z+5\end{bmatrix}^{-1}$$

$$= \begin{bmatrix} \dfrac{z+5}{(z+2)(z+3)} & \dfrac{1}{(z+2)(z+3)} \\ \dfrac{-6}{(z+2)(z+3)} & \dfrac{z}{(z+2)(z+3)} \end{bmatrix}$$

$$[zI - \Phi]^{-1} z^k = \begin{bmatrix} \dfrac{z^{k+1} + 5z^k}{(z+2)(z+3)} & \dfrac{z^k}{(z+2)(z+3)} \\ \dfrac{-6z^k}{(z+2)(z+3)} & \dfrac{z^{k+1}}{(z+2)(z+3)} \end{bmatrix}$$

$$\mathcal{Z}^{-1}\{[zI - \Phi]^{-1} z^k\} = \begin{bmatrix} 3(-2)^k - 2(-3)^k & (-2)^k - (-3)^k \\ -6[(-2)^k - (-3)^k] & -2(-2)^k + 3(-3)^k \end{bmatrix}$$

$$= \Phi^k,$$

so that the general solution of Equation 2.38 with the given values for x_0, Φ, and Gu becomes

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

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 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*.

2.5.1 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 1.3.2.3 and 1.3.2.4 (pages 12 and 13). 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

$$M(H, F, t_0, t_f) = \int_{t_0}^{t_f} \Phi^{\mathrm{T}}(t) H^{\mathrm{T}}(t) H(t) \Phi(t)\, dt \tag{2.42}$$

for a linear dynamic system with fundamental solution matrix $\Phi(t)$ and measurement sensitivity matrix $H(t)$, defined over the continuous-time interval $t_0 \leq t \leq t_f$. 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 $t_0 \leq t \leq t_{k_f}$ has the general form

$$M(H_k, \Phi_k, 1 \leq k \leq k_f) = \left\{ \sum_{k=1}^{k_f} \left[\prod_{i=0}^{k-1} \Phi_{k-i} \right]^{\mathrm{T}} H_k^{\mathrm{T}} H_k \left[\prod_{i=0}^{k-1} \Phi_{k-i} \right] \right\}, \tag{2.43}$$

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 $0 \leq k \leq k_f$. 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.

2.5.2 Observability of Time-Invariant Systems

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

$$M = \begin{bmatrix} H^{\mathrm{T}} & \Phi^{\mathrm{T}} H^{\mathrm{T}} & (\Phi^{\mathrm{T}})^2 H^{\mathrm{T}} & \dots & (\Phi^{\mathrm{T}})^{n-1} H^{\mathrm{T}} \end{bmatrix} \tag{2.44}$$

for discrete-time systems and

$$M = \begin{bmatrix} H^{\mathrm{T}} & F^{\mathrm{T}} H^{\mathrm{T}} & (F^{\mathrm{T}})^2 H^{\mathrm{T}} & \dots & (F^{\mathrm{T}})^{n-1} H^{\mathrm{T}} \end{bmatrix} \tag{2.45}$$

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 [8] presents a derivation obtained by using properties of the characteristic polynomial of F.

2.5.2.1 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 amount of 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 amount of 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 5) 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 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 2.45, 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 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 2.45, 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 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 2.44, 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 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 2.44, is

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

The system is observable.

2.5.3 Controllability of Time-Invariant Linear Systems

2.5.3.1 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 Kalman as what he called the *duality* and *separability* of the estimation and control problems for linear dynamic systems. Kalman's[10] dual concepts are presented here and in the next subsection, although they are not issues for the estimation problem.

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

$$\dot{x}(t) = Fx(t) + Cu(t), \quad z(t) = Hx(t) + Du(t) \tag{2.46}$$

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 2.46 is controllable if and only if matrix S has n linearly independent columns,

$$S = \begin{bmatrix} C & FC & F^2C & \cdots & F^{n-1}C \end{bmatrix}. \tag{2.47}$$

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

$$x_k = \Phi x_{k-1} + \Gamma u_{k-1}, \tag{2.48}$$

$$z_k = Hx_k + Du_k. \tag{2.49}$$

This system model is considered controllable[11] 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

$$S = \begin{bmatrix} \Gamma & \Phi\Gamma & \Phi^2\Gamma & \cdots & \Phi^{N-1}\Gamma \end{bmatrix} \tag{2.50}$$

having rank n.

Example 2.13 Determine the controllability of Example 2.9. The controllability matrix, using Equation 2.47, is

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

[10]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."

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

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

Example 2.14 Determine the controllability of Example 2.11. The controllability matrix, using Equation 2.50, is

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

The system is not controllable.

2.6 SUMMARY

1. 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. 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, e.g., advance in discrete-time steps.)
3. 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*. 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.
4. In order that the future state of a system may be determinable from its current state and future inputs, the dynamical 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 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*.

5. 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 a time $t_k > t_{k-1}$, Φ_{k-1} is the $n \times n$ *STM* 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.

6. If F and C (or Φ and C) do not depend upon t (or k), then the continuous (or discrete) model is called *time invariant*. Otherwise, the model is *time varying*.
7. 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.

8. For a *homogeneous* system, the STM 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 $t_k > t_{k-1} > t_0$.

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

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

10. 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

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

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.

11. 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.

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 STM 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 also 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 STM 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 STM for continuous time and the solution for the continuous-time system with initial conditions

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

2.9 Use the same general approach as in Example 2.7 (Laplace transform method) for deriving the state-transition matrix for the models used in Examples 2.2, 2.3, and 2.6. (damped harmonic resonator), starting with a linear differential equation model in the form

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

Assume that $u(t) = 1$, a constant disturbing force, and

$$\dot{\delta}(t) = \frac{d\delta}{dt}, \qquad \ddot{\delta}(t) = \frac{d^2\delta}{dt^2}, \quad \zeta = \frac{k_d}{2\sqrt{mk_s}} = \frac{1}{2}, \quad \omega_n = \sqrt{\frac{k_s}{m}} = 1.$$

The transformed parameter ζ is now a unitless damping coefficient and ω_n the "natural" (i.e., undamped) frequency of the resonator.
Transform this model to state-space form in continuous time and then use the Laplace transform to obtain the general solution in terms of a state-transition matrix.

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

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

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

2.11 Determine the controllability and observability of the dynamic system model given below:

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

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

2.12 Derive the STM of the time-varying system

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

2.13 Find the STM for

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

2.14 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 $\Phi(0) = I$?

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

2.16 Derive the formula of Equation 2.42 for the observability matrix of a linear dynamic system model in continuous time. (*Hint*: Use the approach of Example 1.3 for estimating the initial state of a system and Equation 2.27 for the state of a system as a linear function of its initial state and its inputs.)

2.17 Derive the formula of Equation 2.43 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 2.43.)

REFERENCES

[1] E. A. Coddington and N. Levinson, *Theory of Ordinary Differential Equations*, McGraw-Hill, New York, 1955.

[2] R. W. Brockett, *Finite Dimensional Linear Systems*, John Wiley & Sons, Inc., New York, 1970.

[3] 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.

[4] P. M. DeRusso, R. J. Roy, and C. M. Close, *State Variables for Engineers*, John Wiley & Sons, Inc., New York, 1965.

[5] L. K. Timothy and B. E. Bona, *State Space Analysis: an Introduction*, McGraw-Hill, New York, 1968.

[6] E. Evangelisti, Ed., *Controllability and Observability: Lectures given at the Centro Internazionale Matematico Estivo (C. I. M. E.) held in Pontecchio (Bologna), July 1–9 1968*, Springer, Italy, 2010.

[7] 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.

[8] K. Ogata, *State Space Analysis of Control Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1967.

3

PROBABILITY AND EXPECTANCY

Chance, too, which seems to rush along with slack reins, is bridled and governed by law.
Anicius Manlius Severinus Boëthius[1] (~480–525 CE)

3.1 CHAPTER FOCUS

There are useful properties of certain statistical parameters of probability distributions that are the same for all probability distributions. One does not need to assume that the probability distribution is Gaussian or any other specific distribution.

These properties are very useful in Kalman filtering, and this is the essential focus of this chapter. The purpose here is to develop the essential notation and theory behind probability distributions as needed for defining and understanding Kalman filtering, which uses probability distributions defined on n-dimensional real linear spaces or manifolds. The mathematical principles involved are generally applicable to more abstract settings, but that will be the primary interest. The major subtopics are as follows:

Mathematical Foundations. It is assumed that the reader has some familiarity with the mathematical foundations of probability theory, as covered by Billingsley [1], Grinstead and Snell [2], or Papoulis [3], for example. Some of the problems

[1] Boëthius was a Roman official under the Ostrogoth King Theodoric the Great (454–526 CE). The quote is from *De Consolatione Philosophiae,* written in 574 CE, when Boëtius was in prison awaiting trial and eventual execution on charges of treason.

Kalman Filtering: Theory and Practice Using MATLAB®, Fourth Edition.
Mohinder S. Grewal and Angus P. Andrews.
© 2015 John Wiley & Sons, Inc. Published 2015 by John Wiley & Sons, Inc.

at the end of the chapter will test your background knowledge. If you have trouble with them, perhaps you should refresh your knowledge by consulting the references at the end of the chapter or equivalent sources. These sources also define the underlying measure theory, in terms of what is meant by the *union* or *intersection* of measurable sets. The treatment of these underlying concepts in this chapter is heuristic and brief.

Probability Density Functions. It makes all this a bit more transparent if we identify probability distributions on real n-dimensional spaces in terms of their *probability density functions* and then use the ordinary calculus to demonstrate the essential properties of the resulting expectancy operators.

Expectancy, Moments, and Optimal Estimates. The key operator needed for developing the essential formulas is what is called the *expectancy operator*, which turns out to be a *linear functional*—a very powerful mathematical tool for our purposes. It is used for defining the *moments* of probability distributions, and the first and second moments of probability distributions on n-dimensional real space $\mathfrak{R}^n$ can be identified with the essential variables used in Kalman filtering. Furthermore, so long as all transformations of the variables involved are linear, their effects on those moments are expressible in simple formulas that are not dependent on any particulars of the underlying probability distributions beyond their moments. These formulas lead directly to the identification of the estimator with the least-mean-squared error—independent of the specific probability distribution involved. This unshackles least-mean-squared estimation from dependency on any particular error probability distribution.

As a reality check, several probability distributions with different probability density functions are used to demonstrate that the important results do not depend on what particular probability density functions is used—so long as it has definable first and second moments.

Nonlinear Effects. In preparation for Chapter 8 (nonlinear extensions), effects of nonlinear transformations on the expected values (means) and expected squared deviation from the mean (covariances) of probability distributions are also modeled and discussed.

3.2 FOUNDATIONS OF PROBABILITY THEORY

Probabilities are a way of quantifying uncertainty. Probability theory is about mathematical models for doing just that.

Its mathematical foundations started with models for gambling using dice or playing cards, in which the number of possible outcomes is generally manageable with a bit of mental arithmetic. Gamblers were primarily concerned with the odds of different outcomes and the expected long-term winnings from different betting strategies. These were often determinable with a pencil and a paper, given mathematical models appropriate to the game.

But uncertainties can also include such things as the miss distance of a projectile, in which case the possible outcomes are modeled as real numbers. To handle cases

such as this, mathematicians would eventually resort to *probability measures*, which characterize the outcomes of the modeled activities in terms of nonnegative numbers. Here, we will not concern ourselves much with measure theory, except to give assurances that it provides the essential mathematical underpinnings of probability theory, and to give some general notion of what it is and what it does. Readers seeking more formal coverage can find it in References [1–3].

3.2.1 Measure Theory

Probability theory changed in the early twentieth century with the development of *measure theory* as part of the foundations of the integral calculus. The rebuilding of probability on the foundation of measure theory had many contributors [4], with a major role played by Andrey N. Kolmogorov (1903–1987) [5]. The theory may not be particularly intuitive in nature, but it does serve as a foundation for integration and probability theory. Its treatment here will be rather cursory.

Measures are nonnegative functions defined on the so-called *sigma algebras* of "measurable sets" in the domain of a potentially integrable function. Measures are not functions in the ordinary sense, because they do not assign a specific value to each point, but to each "measurable set" of points. The base measurable sets for the ordinary (Riemann or Riemann–Stieltjes) integral of the calculus, for example, are built up from finite intervals, using the length of the interval as its measure. The term *sigma-algebra* refers to the fact that the measurable sets obey certain laws about the measures of the set unions and intersections of measurable sets.[2]

Measure theory does not necessarily change the way calculus is done. It mostly shores up its mathematical foundations and—in the process—defines which functions are integrable.

When we write an integral as

$$\int f(x)\, dx,$$

the $\int$ comes from a stylized "S," used as shorthand for "Sum," and the "dx" represents the measure used in defining the integral. Every real function defined somewhere on the real line has a *domain* (set of real values for which the function is defined) and *range* (set of real values the function assigns to numbers in its domain). The measure used for defining integration essentially defines the resulting classes of integrable functions, but the resulting integral calculus is otherwise much the same.

3.2.2 Probability Measures

The distinguishing feature of probability measures is that the probability measure of the union of all measurable sets is always equal to 1 (unity). That is, the probability

[2]Set theory defines an *algebra* of sets, under the operations of set union and set intersection. The "sigma" refers to the additive (arithmetic) properties of measures of set unions and intersections.

that a random sample being in the domain on which it is defined is 1 (one). Just to hedge a bit, probability theorists might call that outcome "almost certain," and only claim such an outcome would "almost surely" happen.

On the other end of things, the probability that an arrow aimed at a target would strike exactly in the center of the bullseye would be zero (except in fiction). There are just too many other places where the arrow could strike.

The value assigned to a measurable set of points by a probability measure is the probability that a random sampling from the probability distribution would lie in that set of points.

There is not much difference between the mathematical properties of probability measures and ordinary measures, other than the fact that probability integrals of the entire domain must equal 1. Integrals with respect to a probability measure p might use the notation dp or $dp(x)$ in place of the Riemann measure dx, but otherwise integrals involving probability measures look and act much the same as ordinary integrals.

3.2.3 Probability Distributions

A *probability distribution* is defined by a probability measure on the measurable subsets of a set of points $\mathcal{S}$. In general, $\mathcal{S}$ could be any abstract measurable set, but the focus here will be exclusively on n-dimensional real spaces, or on multiply connected sets such circles or surfaces of three-spheres in four-dimensional real spaces (used for representing attitude).

We might, for example, define a probability measure over the surface of Earth (a two-dimensional real manifold embedded in a three-dimensional universe) modeling the likelihood that some object approaching Earth from space would strike there. In this case, however, the integral of that probability measure over the entire surface could be <1, with the leftover probability assigned to the rest of the universe to represent the case that the object would miss Earth altogether.

3.2.3.1 Some Definitions

Probability Spaces Formally, a probability space is defined by a triplet $\{\mathcal{S}, \mathcal{A}, \mathcal{P}\}$, where

$\mathcal{S}$ is called the *set of all possible outcomes* of a random event.

$\mathcal{A}$ is a *sigma-algebra* of subsets of $\mathcal{S}$, meaning that if $\mathcal{A}$ includes sets $A \subset \mathcal{S}$ and $B \subset \mathcal{S}$, then it includes their set union $A \cup B$ and set intersection $A \cap B$. The sigma-algebra $\mathcal{A}$ must also contain $\mathcal{S}$ and its "$\mathcal{S}$-complement" $\mathcal{S} - \mathcal{S} = \varnothing$, the *empty set.*

$\mathcal{P}$ is a *probability measure,* which assigns a nonnegative value to all sets $A \in \mathcal{A}$ and $B \in \mathcal{A}$ such that

- $\mathcal{P}(\varnothing) = 0$.

- $\mathcal{P}(S) = 1$.
- $\mathcal{P}(A \cup B) = \mathcal{P}(A) + \mathcal{P}(B) - \mathcal{P}(A \cap B)$.

We will not be using these definitions and properties, but you should be aware that they are part of the foundations of probability theory.

Random Variables (RV). The alternative terms *variate*, *random variable* (RV), or *stochastic variable* are used to denote possible *outcomes* of a draw from a probability distribution, such as drawing numbered balls in a lottery. These terms do not denote a specific outcome, but the ensemble of possible outcomes and their associated probabilities. If the number of possible outcomes is finite, then each outcome may have a specific probability. However, for probability spaces in which the set of possible outcomes $\mathcal{S}$ is a real n-dimensional vector space or manifold, the probability of drawing a specific point $x \in \mathcal{S}$ is more likely to be zero. This is a rather counter-intuitive concept, but it is a consequence of the vastness of the real numbers.

Realizations of Variates. A specific point $x \in S$ would be called a *realization* of a variate, random variable, stochastic variable, or outcome.

Notation. We will observe the common practice of using an uppercase letter to designate a *variate* X, representing the ensemble of values that might be drawn from a probability distribution and to use the corresponding lowercase letter x to denote a *realization* of that draw. The notation $x \in X$ would then indicate that x is a realization of the variate X.

3.2.4 Probability Density Functions

We are perhaps more familiar with having a probability distribution defined in terms of a nonnegative *probability density function* $p(x) \geq 0$ defined for x in n-dimensional real space $\mathfrak{R}^n$. In this case, the fact that it is a probability measure means that its integral

$$\int_{\mathfrak{R}^n} p(x)\, dx = 1. \tag{3.1}$$

The equivalent probability measure of any measurable subset A of $\mathfrak{R}^n$ would then be defined as the integral of the probability density function p over that subset of $\mathfrak{R}^n$, which is also the probability that a randomly chosen value of x would lie in A:

$$P(x \in A) = \int_A p(x)\, dx. \tag{3.2}$$

In this case, the equivalent probability measure is essentially $p(x)\, dx$, where dx represents the Riemann or Riemann–Stieltjes measure—or any suitable measure, for that matter.

Example 3.1 (Gaussian Probability Density) The probability distribution of the average number of occurrences of a "one" (•) in N successive tosses of a die has what is called a "binomial" distribution. As $N \to \infty$, this tends toward a particular type

of distribution called a "*Gaussian*" or "*normal*" distribution. The Gaussian/normal distribution[3] is the limit of many other 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.

The *multivariate* Gaussian probability density function for a column vector x of n components has the form

$$p(x) = \frac{1}{\sqrt{2\pi \det P_{xx}}} \exp\left(-\frac{1}{2}(x-\mu_x)^T P_{xx}^{-1}(x-\mu_x)\right)$$
$$x \in \mathcal{N}(\mu_x,\ \mathcal{P}_{xx}), \tag{3.3}$$

with parameters

μ_x, the *mean* of the distribution, a column vector with n components, and
P_{xx}, the *covariance* of the distribution, an $n \times n$ symmetric positive-definite matrix.

The "$\mathcal{N}$" notation stands for "normal." However, so many things in mathematics are called "normal," it is called "Gaussian" here to avoid ambiguity.

General Probability Integrals A probability measure defines which functions are integrable under that probability measure. In that case, the integral of an integrable function $f(x)$ over a measurable set A could be denoted as

$$\int_A f(x)dp(x),$$

where $dp(x)$ denotes the probability measure being used. This notation allows for the possibility that the probability measure cannot be represented in terms of a probability density function.

Example 3.2 (The Dirac Delta Distribution) Not all legitimate probability distributions can be defined by density functions. The Dirac δ "function," for example, is not a true function, but it can be defined as the probability measure which assigns the value 1 to any measurable set containing a specified point (the mean of the resulting probability distribution). This Dirac probability measure has the property that its variance is zero, implying that there is no uncertainty about the value of its variate. (It is the mean of the distribution, almost surely.) However, the probability distributions of interest in Kalman filtering are not very likely to have zero variance, and problems with variances too close to zero can be ill-conditioned for Kalman filtering implementations in finite-precision arithmetic.

[3]It's probability density function has the property that its Fourier transform is also a Gaussian probability density function. 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."

3.2.5 Cumulative Probability Functions

For any probability measure $dp(x)$ defined on the real line, the function $P(x)$ defined as

$$P(x) \stackrel{\text{def}}{=} \int_{-\infty}^{x} d(x) \tag{3.4}$$

is called the *cumulative probability function* (or just *probability function*) of the probability measure.

If—unlike the Dirac distribution—the probability distribution in question has a density function

$$p(x)\ dx = dp(x),$$

then its cumulative probability function

$$P(x) \stackrel{\text{def}}{=} \int_{-\infty}^{x} p(\chi)\ d\chi. \tag{3.5}$$

Example 3.3 (Gaussian Cumulative Probability Function) The probability function for the univariate Gaussian distribution with mean μ and variance σ^2 is defined as

$$P_{\mathcal{N}}(x,\ \mu,\ \sigma) \stackrel{\text{def}}{=} \frac{1}{\sqrt{2\pi\sigma^2}} \int_{-\infty}^{x} \exp\ \left(-\frac{(\chi-\mu)^2}{2\sigma^2}\right)\ d\chi \tag{3.6}$$

$$= \frac{1}{2}\left[1 + \text{erf}\left(\frac{x-\mu}{\sqrt{2\sigma^2}}\right)\right], \tag{3.7}$$

where erf is an analytic function called the "error function" (callable in MATLAB® as `erf`).

3.3 EXPECTANCY

3.3.1 Linear Functionals

Linear functionals are powerful devices in mathematical analysis, and one of them—the expectancy operator—is especially useful in estimation theory.

Any mapping $\mathcal{F}$ from real functions $f(\cdot)$ to real numbers is called a *functional*.

It is called a *linear* functional if, for any two such functions $f(\cdot)$ and $g(\cdot)$ and for any two real numbers a and b,

$$\mathcal{F}[a\, f(\cdot) + b\ g(\cdot)] = a\ \mathcal{F}[f(\cdot)] + b\ \mathcal{F}[b\ g(\cdot)]. \tag{3.8}$$

Linear functionals play a major role in the definition of *generalized functions* such as the Dirac δ-function and probability measures with no corresponding density functions.

3.3.2 Expectancy Operators

3.3.2.1 Expected Values and Probability Densities The *expected value* of any function $f(x)$ of a variate x from a probability distribution with density function $p(\cdot)$ is defined by the operator

$$\underset{x}{\mathrm{E}}\langle f(x)\rangle \stackrel{\text{def}}{=} \int f(x)p(x)\,dx \tag{3.9}$$

provided the integral exists. If so, the result is a real number and the *expectancy operator* is defined as the mapping of functions to their expected values. We could also represent the operator as

$$\underset{x\in X}{\mathrm{E}}\langle\cdot\rangle$$

to indicate the probability distribution of the variate under consideration (X), but there is not much room for it under the E.

The expectancy operator is also a *linear functional* because, for any real numbers a and b and for any integrable functions $f(\cdot)$ and $g(\cdot)$ defined on the domain of the probability density function $p(\cdot)$,

$$\underset{x}{\mathrm{E}}\langle af(x) + bg(x)\rangle = \int [af(x) + bg(x)]\, p(x)\,dx \tag{3.10}$$

$$= \int a\, f(x)\, p(x)\,dx + \int b\, g(x)\, p(x)\,dx \tag{3.11}$$

$$= a \int f(x)\, p(x)\,dx + b \int g(x)\, p(x)\,dx \tag{3.12}$$

$$= a\ \underset{x}{\mathrm{E}}\langle f(x)\rangle + b\ \underset{x}{\mathrm{E}}\langle g(x)\rangle. \tag{3.13}$$

This result also applies to cases where x and the functions $f(\cdot)$ and $g(\cdot)$ have vector values.

3.3.2.2 Expected Values and Probability Measures The expectancy operator has been defined in terms of a probability density function $p(x),\ x \in X$. However, it can as well be defined in terms of the probability measure of X, without reference to a probability density function.

A probability measure $\wp(\cdot)$ assigns a nonnegative real value to every measurable subset $S \subseteq \mathcal{S}$ in a sigma-algebra of subsets of a parent set $\mathcal{S}$. The *characteristic function* of a measurable set S is the function

$$f_S\,(x) \stackrel{\text{def}}{=} \begin{cases} 1, & x \in S \\ 0, & x \notin S \end{cases} \tag{3.14}$$

and the associated expectancy operator for that probability measure assigns the value

$$\underset{x}{\mathrm{E}}\langle f_S\,(x)\rangle = \wp\,(S), \tag{3.15}$$

where $\wp\,(S)$ is the probability measure of S.

Example 3.4 (Expected Values and Sampling Functionals) The basic Dirac measure assigns the value 1 (one) to all real measurable sets containing the value 0 (zero). The associated expectancy operator, then, would assign the value

$$\underset{x}{\mathrm{E}}\langle f(x)\rangle = f(0)$$

for every Dirac-measurable function f. By subtracting a more desirable sampling argument x_0, this becomes the sampling "function" (functional) for the value of the function f at x_0:

$$\underset{x}{\mathrm{E}}\langle f(x - x_0)\rangle = f(x_0).$$

Note that this is still a linear functional.

3.3.3 Moments of Distributions

If they exist, the expected values of powers of variates of a scalar probability distribution,

$$\underset{x}{\mathrm{E}}\langle x^N\rangle,$$

are called the "*raw*" *moments* of the distribution. Not all distributions have finite raw moments, however.

Example 3.5 (Cauchy Distribution) The Cauchy distribution[4] is used in risk analysis. In quantum mechanics, it is called the *Lorentz distribution* and used for modeling energy distributions of unstable states.

It has a parametric probability density function of the form

$$p_{\text{Cauchy}}(x,\ m,\ \gamma) \overset{\text{def}}{=} \frac{\gamma/\pi}{\gamma^2 + (x-m)^2}, \tag{3.16}$$

defined for $-\infty < x < +\infty$. The parameter m in this case is both the *mode* (maximum probability density) and *median* (value at which the cumulative probability equals $1/2$) of the distribution. The additional positive parameter γ is called a *scaling parameter*. It behaves somewhat like the standard deviation of the distribution, in that the distribution becomes more spread out as γ increases.

However, the Cauchy distribution has no finite mean (first moment) and variance (second central moment).

If they do exist, these moments are constant parameters of the distribution in question, and their values can provide an alternative definition of the distribution.

The notation ${}^{[N]}\mu_x$ will be used to denote the Nth raw moment of a distribution with the variate X—if the moment in question exists.

[4]Named after French mathematician Baron Augustin-Louis Cauchy (1789–1857).

There will always be a *zeroth-order moment*, the scalar

$$ {}^{[0]}\mu_x \stackrel{\text{def}}{=} \underset{x}{\mathrm{E}}\langle x^0 \rangle = \underset{x}{\mathrm{E}}\langle 1 \rangle = \int p_x(x)\, dx = 1, \tag{3.17} $$

because the underlying measure of integration is a probability measure. The zeroth-order moment is always a scalar, independent of the dimension of the variate X involved.

3.3.3.1 Means If it does exists, the first-order moment of the n-dimensional variate X of a multivariate probability distribution will be a vector of the same dimension,

$$ {}^{[1]}\mu_x \stackrel{\text{def}}{=} \underset{x}{\mathrm{E}}\langle x \rangle \stackrel{\text{def}}{=} \begin{bmatrix} \underset{x}{\mathrm{E}}\langle x_1 \rangle \\ \underset{x}{\mathrm{E}}\langle x_2 \rangle \\ \underset{x}{\mathrm{E}}\langle x_3 \rangle \\ \vdots \\ \underset{x}{\mathrm{E}}\langle x_n \rangle \end{bmatrix}, \tag{3.18} $$

called the *mean* of the distribution of X.

3.3.3.2 Central Moments After the mean, the moment of first order ($N = 1$), there are two choices for defining the higher order moments, and the dimensionality of the data structures representing moments of order $N > 1$ depends on N and the dimensionality of the underlying variate X.

Besides the *raw moment* used for defining the mean, there are *central moments* defining the moments "about the mean." For a scalar variate X, the moments of order $N > 1$ would be defined as

$$ {}^{[N]}\mu_x \stackrel{\text{def}}{=} E_x\langle x^N \rangle \qquad \text{(raw moment)} \tag{3.19} $$

$$ {}^{[N]}\sigma_x \stackrel{\text{def}}{=} E_x\langle (x - {}^{[1]}\mu_x)^N \rangle \quad \text{(central moment).} \tag{3.20} $$

Here, the notation ${}^{[N]}\sigma_x$ is used to distinguish the Nth central moment from the raw moment, ${}^{[N]}\mu_x$.

3.3.3.3 Covariance Matrices After the mean ($N = 1$), the next higher order *central moment* ($N = 2$) of an n-dimensional vector variate X is an $n \times n$ matrix

$$ P_{xx} \stackrel{\text{def}}{=} {}^{[2]}\sigma_x \tag{3.21} $$

$$ = \underset{x}{\mathrm{E}}\langle (x - {}^{[1]}\mu_x)(x - {}^{[1]}\mu_x)^{\mathrm{T}} \rangle \tag{3.22} $$

$$= \begin{bmatrix} p_{11} & p_{12} & p_{13} & \cdots & p_{1n} \\ p_{21} & p_{22} & p_{23} & \cdots & p_{2n} \\ p_{31} & p_{32} & p_{33} & \cdots & p_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ p_{n1} & p_{n2} & p_{n3} & \cdots & p_{nn} \end{bmatrix} \tag{3.23}$$

$$p_{ij} \stackrel{\text{def}}{=} E_x\langle (x_i - {}^{[1]}\mu_{x\ i})(x_j - {}^{[1]}\mu_{x\ j})\rangle. \tag{3.24}$$

The equivalent second-order *raw moment* is also an $n \times n$ matrix,

$${}^{[2]}\mu_x = \underset{x}{\mathrm{E}}\langle xx^{\mathrm{T}}\rangle \tag{3.25}$$

$$= \begin{bmatrix} {}^{[2]}\mu_{x\ 11} & {}^{[2]}\mu_{x\ 12} & {}^{[2]}\mu_{x\ 13} & \cdots & {}^{[2]}\mu_{x\ 1n} \\ {}^{[2]}\mu_{x\ 21} & {}^{[2]}\mu_{x\ 22} & {}^{[2]}\mu_{x\ 23} & \cdots & {}^{[2]}\mu_{x\ 2n} \\ {}^{[2]}\mu_{x\ 31} & {}^{[2]}\mu_{x\ 32} & {}^{[2]}\mu_{x\ 33} & \cdots & {}^{[2]}\mu_{x\ 3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ {}^{[2]}\mu_{x\ n1} & {}^{[2]}\mu_{x\ n2} & {}^{[2]}\mu_{x\ n3} & \cdots & {}^{[2]}\mu_{x\ nn} \end{bmatrix} \tag{3.26}$$

$${}^{[2]}\mu_{x\ ij} \stackrel{\text{def}}{=} E_x\langle x_i x_j\rangle. \tag{3.27}$$

3.3.3.4 Moments of Order $N > 2$ For scalar variates X, these are scalars defined by Equations 3.19 (raw moments) and 3.20 (central moments).

When the variates X are n-dimensional vectors, it is a bit more complicated. The general Nth-order moment will be a data structure with dimensions

$$\underbrace{n \times n \times n \times \cdots \times n}_{N \text{ times}}.$$

Any Nth-order raw moment ${}^{[N]}\mu_x$ is a data structure indexed over N subscripts $i_1, i_2, i_3, \ldots, i_N$, with the jth subscript $1 \le i_j \le n$ and the corresponding data structure elements

$${}^{[N]}\mu_{x i_1, i_2, i_3, \ldots, i_N} \stackrel{\text{def}}{=} \underset{x}{\mathrm{E}}\langle x_{i_1} \times x_{i_2} \times x_{i_3} \times \cdots \times x_{i_N}\rangle, \tag{3.28}$$

The corresponding Nth-order central moment ${}^{[N]}\sigma_x$ would have $(x_{i_j} - {}^{[1]}\mu_{x\ i_j})$ in place of x_{i_j} in Equation 3.28.

Example 3.6 (Moments of Univariate Gaussian Distributions) If the distribution in question is univariate Gaussian with mean μ and variance σ^2, the higher order

central moments are all determinable from μ and σ as

$$\sigma_k = \begin{cases} 0, & k \text{ odd}, \\ \sigma^k \ (k-1)\,!!, & k \text{ even}, \end{cases} \tag{3.29}$$

$$(k-1)!! \stackrel{\text{def}}{=} 1 \cdot 3 \cdot 5 \cdot 7 \cdot \cdots \cdot (k-1), \tag{3.30}$$

the so-called "double factorial" for odd numbers.

The higher order raw moments are also determinable from μ and σ, but the relationships are a bit more complex:

$$\mu_k = \begin{cases} \pi^{-1/2} 2^{(k+1)/2} \mu \sigma^{k-1} \Gamma\left(\dfrac{k+1}{2}\right) K\left(\dfrac{1-k}{2}, \dfrac{1}{2}, \dfrac{-\mu^2}{2\sigma^2}\right) & k \text{ odd}, \\ \pi^{-1/2} 2^{k/2} \sigma^k \Gamma\left(\dfrac{k+3}{2}\right) K\left(\dfrac{-k}{2}, \dfrac{3}{2}, \dfrac{-\mu^2}{2\sigma^2}\right) & k \text{ even}, \end{cases} \tag{3.31}$$

where the Gamma function is the positive-real-valued extension of the factorial function,

$$\Gamma(z) \stackrel{\text{def}}{=} \int_0^{+\infty} s^{z-1} e^{-s} \, ds, \tag{3.32}$$

Kummer's confluent hypergeometric function ${}_1F_1$

$$K(a,\ b,\ c) \stackrel{\text{def}}{=} \sum_{k=0}^{+\infty} \frac{(a)_k \ c^k}{(b)_k \ k!}, \tag{3.33}$$

and the *Pochammer notation*

$$(y)_k \stackrel{\text{def}}{=} \frac{\Gamma(y+k)}{\Gamma(y)}. \tag{3.34}$$

Evaluations of Equations 3.31 and 3.29 should produce the values shown in Table 3.1. Unlike the case with raw moments, all odd-order central moments are zero, and all even-order central moments depend only on the second-order central moment.

3.3.3.5 Cross-Covariance Let p_{ij} be the element in the ith row and jth of a covariance matrix P_{xx} of a vector variate X,

$$p_{ij} = \underset{x}{\mathrm{E}}\langle (x_i - \mu_i)(x_j - \mu_j) \rangle. \tag{3.35}$$

p_{ij} is called the *cross-covariance*[5] of the ith and jth components of X.

[5]Strictly speaking, if $i = j$, it is just the *variance* of the ith $= j$th component of X. Only if $i \neq j$, is it called the *cross-covariance* of the scalar variates X_1 and X_j.

TABLE 3.1 Scalar Gaussian Moments

Order N	Central Moment	Raw Moment
0	1	0
1	0	μ
2	σ^2	$\mu^2 + \sigma^2$
3	0	$\mu^3 + 3\mu\sigma^2$
4	$3\sigma^4$	$\mu^4 + 6\mu^2\sigma^2 + 3\sigma^4$
5	0	$\mu^5 + 10\mu^3\sigma^2 + 15\mu\sigma^4$
6	$15\sigma^6$	$\mu^6 + 15\mu^4\sigma^2 + 45\mu^2\sigma^4 + 15\sigma^6$
7	0	$\mu^7 + 21\mu^5\sigma^2 + 105\mu^3\sigma^4 + 105\mu\sigma^6$
8	$105\sigma^8$	$\mu^8 + 28\mu^6\sigma^2 + 210\mu^4\sigma^4 + 420\mu^2\sigma^6 + 105\sigma^8$
9	0	$\mu^9 + 36\mu^7\sigma^2 + 378\mu^5\sigma^4 + 1260\mu^3\sigma^6 + 945\mu\sigma^8$
10	$945\sigma^{10}$	$\mu^{10} + 45\mu^8\sigma^2 + 630\mu^6\sigma^4 + 3150\mu^4\sigma^6 + 4725\mu^2\sigma^8 + 945\sigma^{10}$

More generally, if X_a and X_b represent nonoverlapping subvectors of the vector variate X, then the submatrix

$$P_{ab} = \mathop{\mathrm{E}}_{x}\langle x_a x_b^{\mathrm{T}}\rangle \tag{3.36}$$

of P_{xx} represents the cross-covariance of X_a and X_b.

For example, if X is partitioned into the subvectors

$$x = \begin{bmatrix} X_a \\ X_b \end{bmatrix}, \tag{3.37}$$

then the corresponding covariance matrix can be partitioned as

$$P_{xx} = \begin{bmatrix} P_{x_a x_a} & P_{x_a x_b} \\ P_{x_b x_a} & P_{x_b x_b} \end{bmatrix}, \tag{3.38}$$

where

$P_{x_a x_a}$ is the covariance of X_a,
$P_{x_b x_b}$ is the covariance of X_b,
$P_{x_a x_b}$ is the cross-covariance of X_a and X_b, and
$P_{x_b x_a}$ is the cross-covariance of X_b and X_a.

In general, however,

$$P_{x_a x_b} = P_{x_b x_a}^{\mathrm{T}}. \tag{3.39}$$

3.3.3.6 Correlation Coefficients For any term p_{ij} in the ith row and jth column of a covariance matrix P_{xx}, the ratio

$$\rho_{ij} \stackrel{\text{def}}{=} \frac{p_{ij}}{\sqrt{p_{ii}p_{jj}}} \tag{3.40}$$

is called the *correlation coefficient* between the ith and jth components of X. It has the following properties:

1. $\rho_{ii} = 1$ for all i.
2. $\rho_{ij} = \rho_{ji}$.
3. $-1 \leq \rho_{ij} \leq 1$ for all i and j.

The first two of these assertions are consequences of the definition of ρ_{ij} in Equation 3.40. The last assertion is a consequence of Hölder's inequality, originally defined for vectors [6], but generalized to functions.

Given the values of ρ_{ij} for a covariance matrix P_{xx}, it can be decomposed as a matrix product

$$P_{xx} = \text{diag}(\sigma) C_\rho \text{ diag } (\sigma) \tag{3.41}$$

$$\text{diag } (\sigma) \stackrel{\text{def}}{=} \begin{bmatrix} \sigma_1 & 0 & 0 & \cdots & 0 \\ 0 & \sigma_2 & 0 & \cdots & 0 \\ 0 & 0 & \sigma_1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \sigma_n \end{bmatrix} \tag{3.42}$$

$$\sigma_i \stackrel{\text{def}}{=} \sqrt{p_{ii}}, i = 1, 2, 3, \ldots, n \tag{3.43}$$

$$C_\rho \stackrel{\text{def}}{=} \begin{bmatrix} 1 & \rho_{12} & \rho_{13} & \cdots & \rho_{1n} \\ \rho_{21} & 1 & \rho_{23} & \cdots & \rho_{2n} \\ \rho_{31} & \rho_{32} & 1 & \cdots & \rho_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho_{n1} & \rho_{n2} & \rho_{n3} & \cdots & 1 \end{bmatrix}. \tag{3.44}$$

3.3.3.7 Statistical Independence and Correlation Two variates X and Y are said to be *statistically independent* if their *joint probability density function*

$$p(x, y) = p(x)\, p(y), \tag{3.45}$$

where $p(x)$ and $p(y)$ are the product of the individual probability densities of X and Y, respectively.

For example, if the ith and jth components of a vector variate X are statistically independent, then

$$p(x_i, x_j) = p(x_i)\ p(x_j), \tag{3.46}$$

and their cross-covariance

$$\begin{aligned} p_{ij} &\overset{\text{def}}{=} E_x\langle (x_i - \mu_i)(x_j - \mu_j)\rangle && (3.47)\\ &\overset{\text{def}}{=} \int_{x_i}\int_{x_j} (x_i - \mu_i)(x_j - \mu_j)\ p(x_i, x_j)\ dx_i\ dx_j && (3.48)\\ &= \left[\int_{x_i} \left(x_i - \mu_i\right)\ p(x_i)\ dx_i\right]\ \left[\int_{x_j} \left(x_j - \mu_j\right) p(x_j)\ dx_j\right] && (3.49)\\ &= [\mu_i - \mu_i]\ [\mu_j - \mu_j] && (3.50)\\ &= 0. && (3.51) \end{aligned}$$

That is, statistical independence of components of a vector variate X is equivalent to having zero cross-covariance—or, equivalently, zero correlation coefficient.

The same applies to statistical independence of nonoverlapping subvectors $X_a,\ X_b$ of a vector variate X. In that case, the corresponding submatrix $P_{x_a\ x_b}$ of the covariance matrix P_{xx} will be a zero matrix.

For example, if the vector variate X is partitioned into the statistically independent sub-vector variates X_a and X_b such that the sample vectors

$$x = \begin{bmatrix} x_a \\ x_b \end{bmatrix}, \tag{3.52}$$

then the corresponding covariance matrix can be partitioned as

$$P_{xx} = \begin{bmatrix} P_{x_a x_a} & 0 \\ 0 & P_{x_b x_b} \end{bmatrix}, \tag{3.53}$$

where $P_{x_a x_a}$ is the covariance of X_a and $P_{x_b x_b}$ is the covariance of X_b.

3.4 LEAST-MEAN-SQUARE ESTIMATE (LMSE)

The word *estimate* comes from the Latin verb *aestimare,* meaning "to place a value on (something)." In the present context, that "value" will be a real number or real vector and the "something" will be a probability distribution defined over an n-dimensional real space by its mean and covariance.

Among the more profound discoveries in probability theory are the facts that

1. The mean of a probability distribution defined on $\mathfrak{R}^n$ is also the estimated value that achieves the least-mean-squared estimation error.
2. The *trace* (sum of diagonal elements) of the associated covariance matrix equals that least (or minimum) mean-squared estimation error.

3.4.1 Squared Estimation Error

An estimate has also been called *an educated guess*. Before an archer releases an arrow aimed at a target, her or his best estimate of where it will end up is in the center of the target—or as close thereto as possible.[6] The associated "estimation error" in this case is the miss distance, which is the square root of the sum of the squares of the arrow's horizontal and vertical displacements from the exact center of the target. Minimizing the miss distance in this case is equivalent to minimizing the sum of the squares of the miss vector components. In practice, the likelihood of making that error zero all the time is negligible, so the long-term goal of most archers is to minimize the expected squared miss distance.

By using the expectancy operator, that same goal can be put in terms of probability distributions in a way that defines the optimal estimate $\hat{\mathrm{x}}$ of a value drawn from that probability distribution in terms that depend only on the first moment of the distribution, but not otherwise on the shape of the associated probability density function.

3.4.2 Minimization

Optimal estimates are defined by some optimality criterion. In the case that criterion is the *expected* (i.e., *mean*) squared estimation error, the optimal estimate is the one which minimizes the mean-square estimation error. In that case, it is called the *minimum-mean-squared estimate* (*MMSE*), or *least-mean-squared estimate* (*LMSE*).

For any estimate $\hat{x}$ for the value of x in a probability distribution over n-dimensional real space with first- and second-order moments, the mean-squared estimation error can be defined as

$$\varepsilon^2(\hat{x}) = \mathop{\mathrm{E}}_{x}\langle|\hat{x} - x|^2\rangle \tag{3.54}$$

$$= \mathop{\mathrm{E}}_{x}\langle(\hat{x} - x)^{\mathrm{T}}(\hat{x} - x)\rangle \tag{3.55}$$

$$= \mathop{\mathrm{E}}_{x}\langle\hat{x}^{\mathrm{T}}\hat{x} - 2\hat{x}^{\mathrm{T}}x + x^{\mathrm{T}}x\rangle \tag{3.56}$$

$$= \mathop{\mathrm{E}}_{x}\langle\hat{x}^{\mathrm{T}}\hat{x}\rangle - 2\mathop{\mathrm{E}}_{x}\langle\hat{x}^{\mathrm{T}}x\rangle + \mathop{\mathrm{E}}_{x}\langle x^{\mathrm{T}}x\rangle \tag{3.57}$$

[6] The Greek word for this guessing/aiming process is "στόχος," which gives us the modern word *stochastic*.

$$= \mathop{\mathrm{E}}_{x}\left\langle \sum_j \hat{x}_j^2 \right\rangle - 2\mathop{\mathrm{E}}_{x}\left\langle \sum_j \hat{x}_j x_j \right\rangle + \mathop{\mathrm{E}}_{x}\left\langle \sum_j x_j^2 \right\rangle \tag{3.58}$$

$$= |\hat{x}|^2 - 2\hat{x}^{\mathrm{T}} \mathop{\mathrm{E}}_{x}\langle x\rangle + \mathop{\mathrm{E}}_{x}\langle |x|^2\rangle. \tag{3.59}$$

This mean-squared error $\varepsilon^2(\hat{x}) \to +\infty$ as any component $\hat{x}_j \to \pm\infty$ and achieves its minimum $\hat{x}_{\mathrm{LMSE}}$ where

$$0 = \frac{\partial \varepsilon^2(\hat{x})}{\partial \hat{x}} \tag{3.60}$$

$$= 2\hat{x} - 2\mathop{\mathrm{E}}_{x}\langle x\rangle, \tag{3.61}$$

which can be solved for the estimate $\hat{x}_{\mathrm{LMSE}}$ with the least-mean-squared error,

$$\hat{x}_{\mathrm{LMSE}} = \mathop{\mathrm{E}}_{x}\langle x\rangle. \tag{3.62}$$

That is, the LMSE of x is the mean of the distribution—independent of the probability distribution involved.

3.4.3 Least-Mean-Squared Estimation Error

The covariance of least-mean-squared estimation error is the second central moment of the probability distribution, also known as the *covariance of the probability distribution*,

$$\mathop{\mathrm{E}}_{x}\langle (\hat{x}_{\mathrm{LMSE}} - x)(\hat{x}_{\mathrm{LMSE}} - x)^{\mathrm{T}}\rangle = P_{xx}. \tag{3.63}$$

Equation 3.55 for the squared estimation error, evaluated at $\hat{x} = \mu$, is

$$\varepsilon^2(\mu) = \sum_j \mathop{\mathrm{E}}_{x}\langle (x - \mu)^2\rangle \tag{3.64}$$

$$= \sum_j p_{jj} \tag{3.65}$$

$$= \mathrm{tr}\ (P_{xx}), \tag{3.66}$$

the matrix *trace* (sum of diagonal elements) of P_{xx}.

That is, the mean μ and covariance P_{xx} of any probability distribution characterize the LMSE, in terms of its value and its mean-squared error.

Example 3.7 (Evaluation Using Five Different Distributions) This example computes and plots the mean-square estimation error as a function of the estimated value for five different probability density functions, all defined on the real line with the same means and variances, as plotted in Figure 3.1. These include the

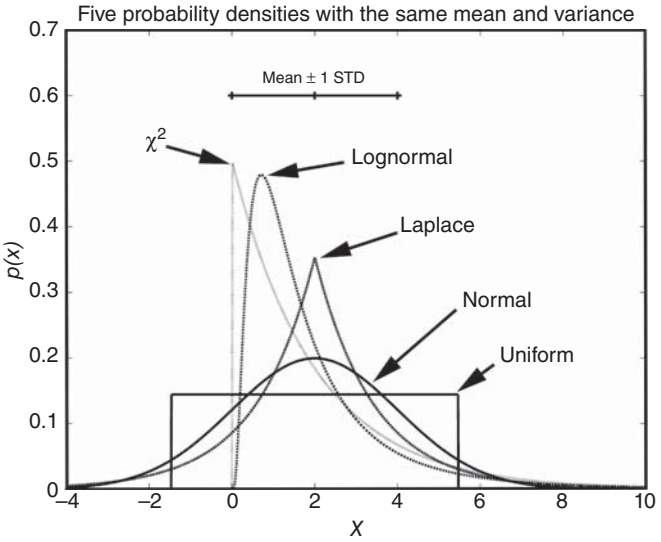


Figure 3.1 Five probability density functions with the same means and variances.

following:

1. A uniform distribution with mean equal to 2 and variance equal to 4. This has zero probability off a fixed interval on the real line. It is used to represent quantization errors in digitized signals, for example.
2. A Gaussian (or normal) distribution with mean equal to 2 and variance equal to 4. This has positive probability everywhere on the real line. The Gaussian distribution is perhaps overused a bit.
3. A Laplace distribution with mean equal to 2 and variance equal to 4. This also has positive probability everywhere on the real line but is different from the Gaussian distribution.
4. A lognormal distribution with mean equal to 2 and variance equal to 4. This is nonzero only for nonnegative input values, and has a "high tail," meaning that it converges to zero as the variate $x \to +\infty$ more slowly than an exponential distribution such as the Gaussian or Laplace distribution. The lognormal distribution is often used to represent the distribution of income or wealth in populations, for example.
5. A chi-squared distribution with two degrees of freedom, which has mean equal to 2 and variance equal to 4. This also has a "thick tail," but not a "high tail," meaning that it also converges slowly to zero as its argument approaches infinity—but not more slowly than exponential distributions. The chi-squared distribution is used in statistical hypothesis testing and for monitoring the behavior of Kalman filters.

The values of the mean-squared estimation error as a function of the value of the estimate are plotted in Figure 3.2, for all five of the distributions. These do not show

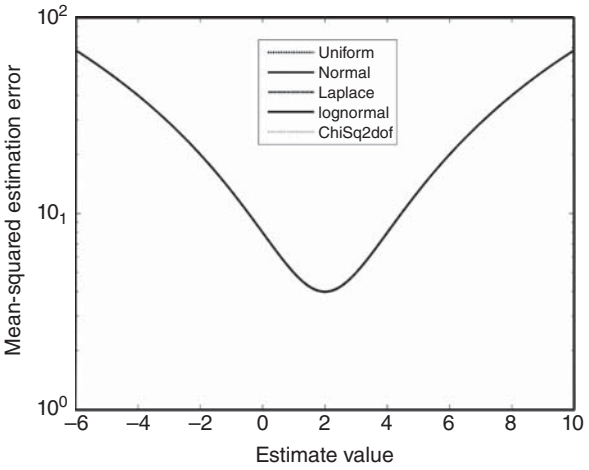


Figure 3.2 Mean-squared estimation error versus estimate value.

much difference between distributions, but the program `meansqesterr.m` (among the m-files on the Wiley web site) generated using MATLAB software also plots the differences—just to show that they are not exactly the same.

The mean-squared-error plots in Figure 3.2 all have the minimum mean-squared-error value of 4—the second central moment of all the distributions—at the estimated value of $\hat{x} = 2$—the mean of all the distributions.

The point is that the LMSE of the variate in all cases is the mean of the distribution, and the mean-squared estimation error is equal to the variance of the distribution—independent of the shape of the distribution.

3.4.4 Means and Covariances: Moments to Remember

Gamblers and actuaries are rightfully concerned about probability distributions, but distributions are not essential for least-mean-squares estimation. That is fortunate, because it is usually much easier and more efficient to estimate just the mean and covariance of a variate x.

3.4.4.1 Recursive Estimates of Means and Covariances The means and covariances of given data sequences $\{x[k] \mid k = 0,\ 1,\ 2,\ \dots\}$ can be estimated recursively by the formulas

$$\hat{\mu}[k+1] = \hat{\mu}[k] + \frac{1}{k+1}[x[k+1] - \hat{\mu}_k] \tag{3.67}$$

$$\widehat{P}_{xx}[k+1] = \left(1 - \frac{1}{k}\right)\widehat{P}_{xx}[k] + (1+k)(\hat{\mu}[k+1] - \hat{\mu}[k])(\hat{\mu}[k+1] - \hat{\mu}[k])^{\mathrm{T}}$$

$$k = 0,\ 1,\ 2,\ 3,\ \dots, \tag{3.68}$$

where $\hat{\mu}[k]$ is the kth recursive estimate of the mean and $\widehat{P}_{xx}[k]$ is the kth recursive estimate of the covariance, using the samples up to the kth element. To start the algorithm off, the value of $\hat{\mu}[0]$ can be initialized at $x[0]$, and the value of $\widehat{P}_{xx}[0]$ can be initialized as the zero matrix.

3.4.4.2 Means and Variances in Least-squares Sensor Calibration Least-squares estimation is used for removing known trends from sensor data, but the same solution can also provide an estimate of the variance of measurement or sensor error. This is also a by-product of the procedure for calibrating a sensor by observing its outputs while controlling its inputs. Such variances are a direct by-product from solving the associated least-squares problem using augmented upper-triangular[7] Cholesky's decomposition. In this approach, the linear system $Ax = b$ is solved by upper-triangular Choleskys decomposition of the augmented matrix symmetric product

$$\left[A \mid b\right]^{\mathrm{T}}[A \mid b] = \left[\begin{array}{c|c} A^{\mathrm{T}}A & A^{\mathrm{T}}b \\ \hline \left(A^{\mathrm{T}}b\right)^{\mathrm{T}} & b^{\mathrm{T}}b \end{array}\right], \tag{3.69}$$

the result of which is an augmented upper-triangular *Choleskys factor* matrix $\mathcal{U}$ with partitioning

$$\mathcal{U} = \left[\begin{array}{c|c} U & y \\ \hline 0 & \varepsilon \end{array}\right], \tag{3.70}$$

where U is upper triangular and the symmetric product

$$\left[\begin{array}{c|c} A^{\mathrm{T}}A & A^{\mathrm{T}}b \\ \hline \left(A^{\mathrm{T}}b\right)^{\mathrm{T}} & b^{\mathrm{T}}b \end{array}\right] = \mathcal{U}^{\mathrm{T}}\mathcal{U} \tag{3.71}$$

$$= \left[\begin{array}{c|c} U^{\mathrm{T}} & 0 \\ \hline 0 & \varepsilon \end{array}\right]^{\mathrm{T}} \left[\begin{array}{c|c} U & y \\ \hline 0 & \varepsilon \end{array}\right] \tag{3.72}$$

$$= \left[\begin{array}{c|c} U^{\mathrm{T}}U & U^{\mathrm{T}}y \\ \hline \left(U^{\mathrm{T}}y\right)^{\mathrm{T}} & |y|^2 + \varepsilon^2 \end{array}\right], \tag{3.73}$$

the last line of which is a matrix equation which can be solved for the least-squares solution $\hat{x}$ as the solution of the Cholesky form of the *normal equation* for the least-squares problem:

$$U\hat{x} = y. \tag{3.74}$$

This avoids the problem of inverting a Gramian matrix, but adds the computational cost of the Cholesky decomposition. The Cholesky form of the normal equation can

[7]Most treatments of Cholesky's decomposition assume a lower-triangular result. However, the upper-triangular equivalent is just as efficient, accurate, and useful. It only means that the resulting back-substitution algorithm for inverting the result will start with the last row and progress backward.

be solved without matrix inversion—by using back substitution, instead—because U is triangular.

Furthermore, the yet-unused result $\varepsilon = |Ax - b|$, the root-sum-squared estimation error. In this case, the unbiased estimate for the standard deviation of the error in the data b would be $\sigma = \varepsilon/\sqrt{n-1}$, where n is the dimension of b. The mean error in the data is called *sensor bias.* It is usually part of the estimated vector $\hat{x}$.

This approach applies to trend removal by fitting input–output pairs of data with a known parametric functional form linear in the unknown parameters, such as fitting b to polynomials in a known input quantity χ. In this case, the elements of A are powers of the input variable χ corresponding to the output values b and the components of the vector x are the unknown coefficients of the polynomial.

3.4.5 Alternative Measures of Miss Distance

A *metric* is a measure of distance between two points in a topological space, which includes n-dimensional real spaces.

The Kalman filter uses the so-called "Euclidean metric," which is the square root of the sum of the squares of the components of the actual impact point relative to the intended target. It is based on the *Euclidean norm,* which is the square root of the sum of the squares of a vector. The Euclidean norm is also called the H_2 *norm*, a reference to the so-called "Hölder p-norm" for $p = 2$. In essence, this sort of metric is a norm applied to the difference between two points.

However, there are alternative metrics for $\Re^n$. One of these is the so-called "taxi-cab metric," based on the "sup-norm" or "H_∞ norm," defined as the maximum absolute component of a vector. The H_∞ norm leads to an alternative development of optimal estimation and control, also called *minimax* estimation and control. For the example of the archer, this would only make the bullseyes on the target be squares, not circles. For side-by-side development and comparison of minimax and Kalman filtering, see Reference 8.

3.5 TRANSFORMATIONS OF VARIATES

The critical question is: If a dependent variate Y is defined in terms of an existing variate X by a function f as

$$Y = f(X), \tag{3.75}$$

where the moments of X are known, what are the mean and covariance of Y?

This matters in Kalman filtering, where the variables of interest are the first and second moments of Y. The Kalman filter depends on f being linear, in which case the answer turns out to be straightforward and simple. The formulas for this case are derived in this section, along with formulas for what happens when f is not linear, but sufficiently smooth.

3.5.1 Linear Transformations

We will now put material from Section 3.3.2 in the context of general transformations of variates—without regard for the underlying distributions beyond their means and covariances.

If $x \in \mathfrak{R}^n$ is an n-dimensional real vector variate with probability density function $p(x)$ and A is a matrix conformable[8] for multiplication by x, the dependent variate

$$y = Ax \tag{3.76}$$

will have mean

$$\mu_y = \underset{x}{\mathrm{E}}\langle y \rangle = \int Ax\, p(x)\, dx = A \int x\, p(x)\, dx = A\mu_x \tag{3.77}$$

and covariance

$$P_{yy} = \underset{x}{\mathrm{E}}\langle (y - \mu_y)(y - \mu_y)^{\mathrm{T}} \rangle \tag{3.78}$$

$$= \underset{x}{\mathrm{E}}\langle (Ax - A\mu_x)(Ax - A\mu_x)^{\mathrm{T}} \rangle \tag{3.79}$$

$$= \underset{x}{\mathrm{E}}\langle A(x - \mu_x)(x - \mu_x)^{\mathrm{T}} A^{\mathrm{T}} \rangle \tag{3.80}$$

$$= A \underset{x}{\mathrm{E}}\langle (x - \mu_x)(x - \mu_x)^{\mathrm{T}} \rangle A^{\mathrm{T}} \tag{3.81}$$

$$= AP_{xx}A^{\mathrm{T}}. \tag{3.82}$$

These results do not depend on the details of the probability distributions but only on their first moments and second central moments. That is, any linear transformation of the variates transforms the mean and covariance matrix of the distribution that is exactly the same, independent of the underlying probability distribution.

3.5.1.1 Linear Combinations of Vector Variates Given the definition and properties of cross-covariance and statistical independence, one can uncover some useful properties of linear combinations of vector variates, where the coefficients of the linear combination are now matrices.

If X_a and X_b are vector variates with some as-yet unspecified joint probability distribution, then the composite sample vectors

$$x \overset{\text{def}}{=} \begin{bmatrix} x_a & \in & X_a \\ x_b & \in & X_b \end{bmatrix} \tag{3.83}$$

[8] A notation defined in Appendix B on the companion Wiley web site, and meaning the data structure in question has the correct dimensions for the way it is being used.

will have covariance matrix

$$P_{xx} = \begin{bmatrix} P_{x_a x_a} & P_{x_a x_b} \\ P_{x_b x_a} & P_{x_b x_b} \end{bmatrix}. \tag{3.84}$$

Any linear combination

$$y = A\ x_a + B\ x_b \tag{3.85}$$

$$= \begin{bmatrix} A & B \end{bmatrix} x \tag{3.86}$$

will then have mean

$$\mu_y = A\ \mu_{x_a} + B\ \mu_{x_b} \tag{3.87}$$

and covariance

$$P_{yy} = \begin{bmatrix} A & B \end{bmatrix} \begin{bmatrix} P_{x_a x_a} & P_{x_a x_b} \\ P_{x_b x_a} & P_{x_b x_b} \end{bmatrix} \begin{bmatrix} A & B \end{bmatrix}^{\mathrm{T}} \tag{3.88}$$

$$= AP_{x_a x_a}A^{\mathrm{T}} + AP_{x_a x_b}B^{\mathrm{T}} + BP_{x_b x_a}A^{\mathrm{T}} + BP_{x_b x_b}B^{\mathrm{T}}. \tag{3.89}$$

In the case that X_a and X_b are *statistically independent,*

$$P_{yy} = AP_{x_a x_a}A^{\mathrm{T}} + BP_{x_b x_b}B^{\mathrm{T}}. \tag{3.90}$$

3.5.1.2 Affine Transformations An *affine transformation* is essentially the first two terms (zeroth order and first order) of a series expansion of a function. It is equivalent to a linear transformation plus an offset, such as

$$y = Ax + b, \tag{3.91}$$

where b is a conformable constant vector.

For any variate x with mean μ_x, the mean of y after such an affine transformation will be

$$\mu_y \stackrel{\text{def}}{=} \mathop{\mathrm{E}}_{x}\langle y \rangle \tag{3.92}$$

$$= \mathop{\mathrm{E}}_{x}\langle Ax \rangle + \mathop{\mathrm{E}}_{x}\langle b \rangle \tag{3.93}$$

$$= A \mathop{\mathrm{E}}_{x}\langle x \rangle + b \tag{3.94}$$

$$= A\mu_x + b. \tag{3.95}$$

Similarly, its covariance

$$P_{yy} \stackrel{\text{def}}{=} \mathop{\mathrm{E}}_{x}\langle (y - \mu_y)(y - \mu_y)^{\mathrm{T}} \rangle \tag{3.96}$$

$$= \underset{x}{\mathrm{E}}\langle (Ax + b - A\mu_x - b)(Ax + b - A\mu_x -)^{\mathrm{T}} \rangle \tag{3.97}$$

$$= \underset{x}{\mathrm{E}}\langle A(x - \mu_x)(x - \mu_x)^{\mathrm{T}} A^{\mathrm{T}} \rangle \tag{3.98}$$

$$= AP_{xx}A^{\mathrm{T}}. \tag{3.99}$$

In summary, then, affine transformations add a constant bias to the mean, but otherwise behave the same as linear transformations.

3.5.1.3 Independent Random Offsets Linearity also extends to linear combinations of independent random vector variates.

Let the dependent variate

$$y = Ax + Bw, \tag{3.100}$$

where x is a vector variate with mean μ_x and w is a statistically independent variate with mean μ_w. Then the mean of y

$$\mu_y = \underset{x,w}{\mathrm{E}} \langle y \rangle \tag{3.101}$$

$$= \underset{x}{\mathrm{E}}\langle Ax \rangle + \mathrm{E}_w \langle Bw \rangle \tag{3.102}$$

$$= A \underset{x}{\mathrm{E}}\langle x \rangle + B\mathrm{E}_w \langle w \rangle \tag{3.103}$$

$$= A\mu_x + B\mu_w \tag{3.104}$$

and the covariance

$$P_{yy} = \underset{x,w}{\mathrm{E}} \langle (Ax + Bw - A\mu_x - B\mu_w)(Ax + Bw - A\mu_x - B\mu_w)^{\mathrm{T}} \rangle \tag{3.105}$$

$$= \underset{x}{\mathrm{E}}\langle A(x - \mu_x)(x - \mu_x)^{\mathrm{T}} A^{\mathrm{T}} \rangle + \mathrm{E}_w \langle B(w - \mu_w)(w - \mu_w)^{\mathrm{T}} B^{\mathrm{T}} \rangle \tag{3.106}$$

$$= AP_{xx}A^{\mathrm{T}} + BP_{ww}B^{\mathrm{T}}. \tag{3.107}$$

This shows the result of adding independent random noise to a vector variate.

3.5.2 Transformations by Analytic Functions

The formulas for linear transformations of vector variates are simple and straightforward. It is not so simple when the transformations are nonlinear.

We consider here the case in which the transformations can be represented in terms of a power series of the variate. This includes the affine case when the coefficients are zero beyond the first-order term in the series, but it also shows how things unravel beyond that.

3.5.2.1 The Scalar Case Scalar analytic functions are defined by power series,

$$f(x) = \sum_{k=0}^{\infty} a_k x^k. \tag{3.108}$$

Transformation of the Mean The expected value

$$\underset{x}{\mathrm{E}}\langle f(x)\rangle = \underset{x}{\mathrm{E}}\left\langle \sum_{k=0}^{\infty} a_k x^k \right\rangle \tag{3.109}$$

$$= \sum_{k=0}^{\infty} a_k \underset{x}{\mathrm{E}}\langle x^k\rangle \tag{3.110}$$

$$= \sum_{k=0}^{\infty} a_k \, {}^{[k]}\mu_x \tag{3.111}$$

$${}^{[k]}\mu_x \overset{\text{def}}{=} \underset{x}{\mathrm{E}}\langle x^k\rangle, \tag{3.112}$$

the kth raw moment of the underlying probability distribution.

Equation 3.111 then defines the LMSE of the value of an analytic function in terms of the coefficients of the power series expansion of the function and the moments of the probability distribution.

This shows that the mean of a variate after an analytic transformation depends on the original moments of all orders, weighted by the coefficients of the power series expansion.

Transformation of Covariance The other moment of interest is the value of the mean-squared estimation error,

$$\underset{x}{\mathrm{E}}\langle (f(x) - \underset{x}{\mathrm{E}}\langle f(x)\rangle)^2\rangle = \sum_{k=0}^{\infty} \left[\sum_{j=0}^{k} a_j^\star a_{k-j}^\star \right] \, {}^{[k]}\mu_x \tag{3.113}$$

$$a_0^\star = a_0 - \sum_{k=0}^{\infty} a_k \, {}^{[k]}\mu_x \tag{3.114}$$

$$a_j^\star = a_j, \; j > 0, \tag{3.115}$$

showing that, in the case of an analytic function, the mean-squared estimation error also involves all orders of moments of the initial probability distribution.

Example 3.8 (Quadratic Transformation of a Scalar Gaussian Variate) Let $x \in X = \mathcal{N}(\mu_x, P_{xx})$ for scalar μ_x and P_{xx}, and let the dependent variate Y be realized as

$$y = y_0 + y_1 x + y_2 x^2,$$

where the scalar constants y_0, y_1, and y_2 are known.

Then the mean of Y

$$\begin{aligned}
\mu_y &= \underset{x}{\mathrm{E}}\langle y_0 + y_1 x + y_2 x^2 \rangle \\
&= y_0 + y_1 \mu_x + y_2 \underset{x}{\mathrm{E}}\langle [(x - \mu_x) + \mu_x]^2 \rangle \\
&= y_0 + y_1 \mu_x + y_2 [\underset{x}{\mathrm{E}}\langle (x - \mu_x)^2 \rangle + 2 \underset{x}{\mathrm{E}}\langle (x - \mu_x)\mu_x \rangle + \mu_x^2] \\
&= y_0 + y_1 \mu_x + y_2 [P_{xx} + \mu_x^2]
\end{aligned}$$

now involves the covariance of X.

Worse yet, the covariance of y

$$\begin{aligned}
P_{yy} &= \underset{x}{\mathrm{E}}\langle (y_0 + y_1 x + y_2 x^2 - \mu_y)^2 \rangle \\
&= \underset{x}{\mathrm{E}}\langle (y_0 + y_1 x + y_2 x^2 - y_0 - y_1 \mu_x - y_2 P_{xx} - y_2 \mu_x^2)^2 \rangle \\
&= \underset{x}{\mathrm{E}}\langle [y_1 (x - \mu_x) + y_2 (x^2 - P_{xx} - \mu_x^2)]^2 \rangle \\
&= y_1^2 \underset{x}{\mathrm{E}}\langle (x - \mu_x)^2 \rangle \\
&\quad + 2 y_1 y_2 \underset{x}{\mathrm{E}}\langle (x - \mu_x)(x^2 - P_{xx} - \mu_x^2) \rangle \\
&\quad + y_2^2 \underset{x}{\mathrm{E}}\langle (x^2 - P_{xx} - \mu_x^2)^2 \rangle \\
&= y_1^2 P_{xx} + 2 y_1 y_2 \underset{x}{\mathrm{E}}\langle (x - \mu_x)[(x - \mu_x)(x + \mu_x) - P_{xx}] \rangle \\
&\quad + y_2^2 E_x \langle [(x - \mu_x)(x + \mu_x) - P_{xx}]^2 \rangle
\end{aligned}$$

now involves moments up to fourth order. One can complete this formula by plugging in all the moments of the original Gaussian distribution, which only depend on μ_x and P_{xx}.

The requisite higher order moments from Table 3.1 could be plugged into the last formula to express everything in terms of the original Gaussian first- and second-order central moments μ and σ^2. However, the resulting distribution will no longer be Gaussian, and the resulting formula using Gaussian moments would not work the next time a nonlinear transformation is applied.

3.5.2.2 The Vector Case An affine transformation includes just the first two terms of a vector-valued power series, the first (zeroth-order) term of which is a constant offset vector ${}^{[0]}a$ and the second (first-order) term of which is a linear transformation,

characterized by a two-dimensional data array (matrix) $^{[1]}A$:

$$f(x) = {}^{[0]}a + {}^{[1]}Ax + \cdots \tag{3.116}$$

$$= \begin{bmatrix} {}^{[0]}a_1 \\ {}^{[0]}a_2 \\ {}^{[0]}a_3 \\ \vdots \\ {}^{[0]}a_n \end{bmatrix} + F_1 x + \cdots , \tag{3.117}$$

with the contribution to the ith component of $f(x)$ being

$$f_i(x) = {}^{[0]}a_i + \sum_{j=1}^{n} {}^{[1]}A_{i,j}x_j + \cdots . \tag{3.118}$$

The next (second-order) term of that series uses a three-dimensional data array $^{[2]}A$, with its contribution to the ith column of $f(x)$ being

$$f_i(x) = F_{0,i} + \sum_{j=1}^{n} F_{1,i,j}x_j + \sum_j \sum_k F_{2,i,j,k}x_j x_k + \cdots . \tag{3.119}$$

The next higher order (third-order) term will have the form

$$f_i(x) = F_{0,i} + \sum_{j=1}^{n} F_{1,i,j}x + j + \sum_j \sum_k F_{2,i,j,k}x_j x_k + \sum_j \sum_k \sum_\ell F_{3,i,j,k,\ell}x_j x_k x_\ell + \cdots , \tag{3.120}$$

and so forth for all higher order terms. Each successive term uses a data array of one dimension greater than the previous term, and the implementation has one more summation. The data requirements just to hold the multidimensional coefficient arrays will then be n^k for the kth term, or

$$\sum_{k=1}^{N} n^k = \begin{cases} N, & n = 1 \\ \dfrac{n\left(n^N - 1\right)}{(n-1)}, & n > 1 \end{cases} \tag{3.121}$$

for the first N terms.

The kth term then requires $\mathcal{O}(k\ n^k)$ arithmetic operations for evaluation (counting the multiplications of powers of the variates), and an expansion to Nth order would require something in the order of

$$\sum_{k=1}^{N} k\ n^k = \begin{cases} N\left(N+1\right)/2, & n = 1 \\ -\dfrac{Nn^{N+1}}{1-n} + \dfrac{n(1-n^N)}{(1-n)^2}, & n > 1, \end{cases} \tag{3.122}$$

arithmetic operations.

Impact on Means and Covariances The down side of all this is that the higher order terms in the transformation require more data arrays in the model, and more central moments of the underlying probability distribution beyond the covariance—just for computing the covariances used in Kalman filtering.

For example, with just one additional (second-order) term in the expansion, the term in the ith row and jth column of the covariance matrix of $f(x)$

$$P_{ff,i,j} = \mathop{\mathrm{E}}_{x}\langle f_i(x) f_j(x)\rangle \tag{3.123}$$

$$= \mathop{\mathrm{E}}_{x}\left\langle \left[F_{0,i} + \sum_{k=1}^{n} F_{1,i,k} x_k + \sum_{k}\sum_{\ell} F_{2,i,k,\ell} x_k x_\ell\right] \times \left[F_{0,j} + \sum_{k=1}^{n} F_{1,j,k} x + j + \sum_{k}\sum_{\ell} F_{2,j,k,\ell} x_k x_\ell\right]\right\rangle, \tag{3.124}$$

which now includes taking the expected value of terms up to fourth order. That is, computing the covariance matrix of a second-order transformation of a variate requires knowing the central moments of its distribution up to fourth order.

In general, computing the covariance matrix of an Nth-order transformation of a variate requires knowing the central moments up to $(2N)$th order. If the resulting distribution were Gaussian, all central moments can be derived from just the mean and covariance. However, nonlinear transformations of Gaussian distributions are no longer Gaussian.

That can be a bit discouraging, when all we want are means and covariance matrices.

3.5.3 Transformation of Probability Density Functions

Kalman filtering is not much concerned with probability density functions, except as an intermediary step for understanding the properties of the expectancy operator $\mathrm{E}x\langle\cdot\rangle$. However, it is sometimes useful to understand how transformations of the variate transform the probability density functions.

3.5.3.1 Linear Transformations Linear transformations of variates defined on the real line $\mathfrak{R}$ are defined by $Y = aX$, where $a \neq 0$. If the variate X has probability density function $p_x(\cdot)$, then the probability density function of Y will be defined by

$$p_y(y) = |a|^{-1} p_x(a^{-1} y). \tag{3.125}$$

3.5.3.2 Nonlinear Transformations If the scalar variate Y is defined by $y = f(x)$, $x \in X$ with probability density $p_x(\cdot)$, the derivation is not so simple as in

the linear case. However, if $f(\cdot)$ and its inverse function $f^{-1}(\cdot)$ are differentiable everywhere, then the probability density function of Y is given by

$$p_y(y) = \left| \frac{\partial f^{-1}(y)}{\partial y} \right| p_x(f^{-1}(y)) \tag{3.126}$$

$$= \frac{p_x(f^{-1}(y))}{\left| \frac{\partial f(x)}{\partial x} \Big|_{x = f^{-1}(y)} \right|}, \tag{3.127}$$

where the absolute value $|\cdot|$ is needed to keep the probability density function $p_y \geq 0$, and this only works where the derivative of $f(\cdot) \neq 0$.

Note that this is equivalent to Equation 3.125 in the case that $f(x) = ax$.

Example 3.9 (Arctangent Transformation of Univariate Gaussian Distribution) Let

$$y = f(x)$$

$$= \arctan\,(a\,x)$$

$$f^{-1}(y) = \frac{1}{a} \tan\,(y),$$

where $x \in X$, a zero-mean unit normal distribution with probability density function

$$p_x(x) = \frac{\exp\,(-x^2/2)}{\sqrt{2\pi}}.$$

Note that both $f(\cdot)$ and $f^{-1}(\cdot)$ are differentiable, and the derivative of $f^{-1}(y) > 0$ and finite for $-\frac{\pi}{2} < y < \frac{\pi}{2}$. Applying Equation 3.126,

$$p_y(y) = \left| \frac{1}{a} \frac{\partial \tan(y)}{\partial y} \right| p_x\left(\frac{1}{a} \tan(y) \right)$$

$$= \frac{1}{a\sqrt{2\pi}} \{1 + [\tan(y)]^2\} \exp\left(-\frac{1}{2} \left[\frac{\tan(y)}{a} \right]^2 \right), \quad -\frac{\pi}{2} < y < \frac{\pi}{2}.$$

It has the shapes shown in Figure 3.3 for various values of the positive parameter a. These might resemble a Gaussian distribution when $a \ll 1$, but are decidedly non-Gaussian and bimodal for $a \gtrapprox 1$.

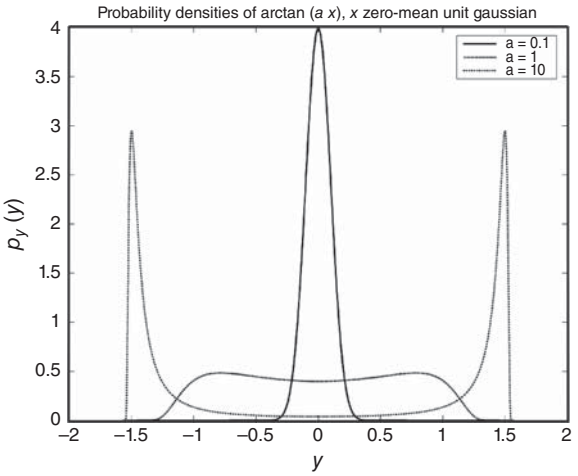


Figure 3.3 Probability densities of $y = \arctan(ax)$, X zero-mean unit-Gaussian.

3.6 THE MATRIX TRACE IN STATISTICS

The *trace* of a square matrix A is defined as the sum of its diagonal elements:

$$\text{tr } [A] \stackrel{\text{def}}{=} \sum_{i=1}^{n} a_{ii}. \tag{3.128}$$

It has some properties found to be useful in statistics—and in Kalman filtering.

3.6.1 Connecting Covariances and Mean-Squared Magnitudes

If P is the covariance (second central moment) of n-vector variate X, then its trace

$$\text{tr } [P] = \sum_{i=1}^{n} p_{ii} \tag{3.129}$$

$$= \sum_{i=1}^{n} \underset{x}{\text{E}}\langle (x_i - \mu_{x_i})^2 \rangle \tag{3.130}$$

$$= \underset{x}{\text{E}}\langle |x - \mu_x|^2 \rangle, \tag{3.131}$$

the mean-squared magnitude of $x - \mu_x$.

3.6.2 A Linear Functional

It we consider an $n \times n$ matrix to be a real function defined on the pairs of integers $\{(i,\ j)|1 \leq i \leq n,\ 1 \leq j \leq n\}$, then the trace becomes a linear functional:

$$\text{tr } [A + B] = \sum_{i=1}^{n} (a_{ii} + b_{ii}) \tag{3.132}$$

$$= \sum_{i=1}^{n} a_{ii} + \sum_{i=1}^{n} b_{ii} \tag{3.133}$$

$$= \text{tr } [A] + \text{tr } [B] \tag{3.134}$$

$$\text{tr } [cA] = \sum_{i=1}^{n} c\ a_{ii} \tag{3.135}$$

$$= c \sum_{i=1}^{n} a_{ii} \tag{3.136}$$

$$= c\ \text{tr } [A]. \tag{3.137}$$

3.6.3 Matrix Product Commutation Under the Trace

If the matrix A is $n \times n$ and the matrix B is $m \times n$, then their products AB and BA are both square matrices. (AB is $n \times n$ and BA is $m \times m$.) Therefore, either product has a matrix trace.

What is even more profound is that their traces are equal:

$$\text{tr } [AB] = \sum_{i=1}^{n} \underbrace{\sum_{j=1}^{m} a_{ij} b_{ji}}_{\{AB\}_{ii}} \tag{3.138}$$

$$= \sum_{j=1}^{m} \underbrace{\sum_{i=1}^{n} b_{ji} a_{ij}}_{\{BA\}_{jj}} \tag{3.139}$$

$$= \text{tr } [BA]. \tag{3.140}$$

That is, the trace of AB equals the trace of BA.

In other words, cyclical permutation of matrix factors under the trace operator can be allowed:

$$\text{tr } [A_1 \times A_2 \times A_3 \times \cdots \times A_N] = \text{tr } [A_2 \times A_3 \times \cdots \times A_N \times A_1] \tag{3.141}$$

$$= \text{tr } [A_N \times A_1 \times A_2 \times A_3 \times \cdots \times A_{N-1}], \tag{3.142}$$

and these formulas can be applied repeatedly to obtain any cyclical permutation of the factors.

As a consequence, if $\nu \in \mathcal{N}(0,\ P_{\nu\nu})$, then the expected value

$$\begin{aligned}
\mathrm{E}_\nu\langle \nu^{\mathrm{T}} P_{\nu\nu}^{-1} \nu\rangle &= \mathrm{E}_\nu\langle \mathrm{tr}\ [\nu^{\mathrm{T}} P_{\nu\nu}^{-1} \nu]\rangle && (3.143)\\
&= \mathrm{E}_\nu\langle \mathrm{tr}\ [P_{\nu\nu}^{-1} \nu\nu^{\mathrm{T}}]\rangle (\text{trace property}) && (3.144)\\
&= \mathrm{tr}\ [P_{\nu\nu}^{-1} \mathrm{E}_\nu\langle \nu\nu^{\mathrm{T}}\rangle] && (3.145)\\
&= \mathrm{tr}\ [P_{\nu\nu}^{-1} P_{\nu\nu}] && (3.146)\\
&= \mathrm{tr}\ [I_\ell] && (3.147)\\
&= \ell, && (3.148)
\end{aligned}$$

the dimension of the variate ν. If properly modeled and the noise is Gaussian, the sequence of values $\{\nu_k^{\mathrm{T}} P_{\nu\nu\ k}^{-1} \nu_k\}$ should have a unit chi-squared distribution with ℓ degrees of freedom. The mean of this distribution is ℓ—as shown above—and the variance is 2ℓ.

3.6.4 Chi-Squared Test

The "chi-squared" (χ^2) test [8] was designed to test whether a given sequence $\{\nu_k\}$ of zero-mean white Gaussian noise is from a Gaussian distribution with a given covariance $P_{\nu\nu}$. If so, the scalar variates

$$\xi(\nu_k) \stackrel{\text{def}}{=} \nu_k^{\mathrm{T}} P_{\nu\nu}^{-1} \nu_k \tag{3.149}$$

should have a unit chi-squared distribution with ℓ degrees of freedom, where ℓ is the dimension of the ν_k.

The probability density function for a unit chi-squared (χ^2) variate $\nu > 0$ with ℓ degrees of freedom is

$$p_{\chi^2}(\nu,\ \ell) = \frac{\nu^{\ell/2-1} e^{-\nu/2}}{2^{\ell/2}\Gamma(\ell/2)}, \tag{3.150}$$

which can be calculated using the MATLAB function `chi2pdf` in the Statistics Toolbox, or using the GNU[9] version with the same name. This MATLAB function computes the sequence of probability densities for a given sequence of samples. The MATLAB function `prod` converts the array of probability densities into a joint probability density for the entire sequence.

Given N such samples of χ^2 variates, this procedure yields the relative probability density function for each of the samples. The statistical decision in this case will be that data sequence with the highest joint probability density.

[9]Distributed by the Free Software Foundation, which claims that "GNU" stands for "GNU's Not Unix."

3.6.5 Schweppe Likelihood Ratio Detection

The chi-square test can also be used when there are more than two possibilities for the distribution behind the observed samples.

Schweppe [9] applied this to the case in which there are two competing hypotheses:

1. A measured signal comes only from a noise source with a known stochastic dynamic model.
2. The measured signal is the sum of the above noise source, plus a signal with known distinct stochastic structure.

In this case, the solution has come to be called *Schweppe Gaussian likelihood ratio detection*. It answers the question of which of two different Gaussian linear stochastic system models ("noise-only" model or a "signal + noise" model) is more likely to be the best model for an observed sequences of sampled values.

It is a form of likelihood ratio test using the linear stochastic models of Kalman filtering. The approach subsumes Kalman filtering, but the test itself only involves testing of white noise processes with one of the two given stochastic models: one with a "noise-only" model and another with a "signal + noise" model. If only noise is present, the sequence of differences between the filter-predicted values and the measured values should be a zero-mean Gaussian white noise sequence

$$\{\nu_k^{[N]} |\ k = 1,\ 2,\ 3,\ \ldots\}$$

with known covariance $P_{\nu\nu}^{[N]}$ (computed by the Kalman filter).

If, on the other hand, the "signal + noise" model is the correct one, the corresponding sequence

$$\{\nu_k^{[S]}\ |\ k = 1,\ 2,\ 3,\ \ldots\}$$

should be a zero-mean Gaussian white noise sequence with known covariance $P_{\nu\nu}^{[S]}$—also computed by its Kalman filter.

Signal detection is then a decision process comparing the relative probabilities of the two sequences of differences between the predicted and observed measurements from two Kalman filters. In either case, if properly modeled, the sequence of outputs should be a zero-mean white (uncorrelated) Gaussian noise process with known covariance. The decision process compares the respective Gaussian probability densities with the assumed covariance matrices for each model. The Gaussian probability density function $p(\nu)$ for $\nu \in \mathcal{N}(0,\ P_{\nu\nu})$ in either case is given by Equation 3.3, and the ratio of the probability densities

$$\frac{p(\nu_k^{[N]})}{p(\nu_k^{[S]})} = \frac{\sqrt{\det P_{\nu\nu}^{[S]}}\exp\ \left(-\frac{1}{2}\nu_k^{[N]\mathrm{T}}\left(P^{[N]}\right)^{-1}\nu_k^{[N]}\right)}{\sqrt{\det P_{\nu\nu}^{[S]}}\exp\ \left(-\frac{1}{2}\psi_i^{\mathrm{T}}\left(P^{[S]}\right)^{-1}\psi_i\right)} \tag{3.151}$$

$$\log\left[\frac{p(\nu_i)}{p(\psi_i)}\right] = -\frac{1}{2}\nu_k^{[N]\mathrm{T}}(P^{[N]})^{-1}\nu_k^{[N]} + \frac{1}{2}\nu_k^{[S]\mathrm{T}}(P^{[S]})^{-1}\nu_k^{[S]} + \frac{1}{2}[\log\det P_{\nu\nu}^{[S]} - \log\det P_{\nu\nu}^{[S]}], \tag{3.152}$$

which can be used to guess which sequence is more likely to have P as its covariance.

If the sequence in question is truly white, the joint probability density of a partial sequence of outputs is the product of the individual probability densities, and logarithm of that joint probability density is the sum of the logarithms of the individual probability densities.

3.6.6 Multihypothesis Detection

This approach can be extended to selecting which of $N > 2$ such statistical sequences is most likely, something which has been used in extended Kalman filtering to overcome linearization errors in the initial (detection) phase of detection and tracking (see Chapter 8).

3.7 SUMMARY

1. Probability theory began with questions about the odds of discrete outcomes (especially in gambling), but progressed to problems with inputs and outputs characterized by real numbers.
2. Probability distributions defined on real domains are defined in terms of probability *measures*—a wholly different breed of cat.
3. Kalman filtering is based on probability distributions defined on n-dimensional real vector spaces. The statistical characteristics of these probabilities defined in this chapter would apply as well to applications defined on closed n-dimensional topological manifolds without boundaries, such as angles defined on the circle or attitudes defined on the three-dimensional surface of the unit sphere in four-dimensional quaternion space. Statistical properties of probability distributions are not an issue with applications defined on $\mathfrak{R}^n$ or on n-dimensional manifolds other than $\mathfrak{R}^n$.
4. The critical variables used in Kalman filtering can be identified with the first two *moments* of the underlying probability distributions.
5. The first moment is called the *mean* of the distribution. It is an n-dimensional vector.
6. The mean is also the *estimate* of the variate with the *least-mean-squared estimation error*. This result does not depend on what probability distribution is used, so long as it has the required first and second moments.
7. The second moment of the deviation from the mean is called the *covariance* of the distribution, also called the *second central moment* of the distribution. It is an $n \times n$ symmetric positive-semi-definite matrix.

8. Subvectors of a vector variate are *statistically independent* if and only if their cross-covariances are zero.
9. The covariance matrix characterizes the *minimum-mean-squared estimation error*.
10. Linear transformations $y = Ax$ of an n-dimensional vector variate x with mean μ_x results in a y probability distribution in which the transformed mean $\mu_y = A\mu_x$—a result which depends only on μ_x and the transforming matrix A and not on any other attributes of the distribution.
11. The covariance matrix of the resulting y-distribution will be $P_{yy} = AP_{xx}A^{\mathrm{T}}$, where P_{xx} is the covariance of the x-distribution.
12. Any linear combination $y = Ax_a + Bx_b$ of vector variates X_a and X_b will have mean $\mu_y = A\mu_{x_a} + B\mu_{x_b}$ and covariance

$$P_{yy} = AP_{x_a x_a}A^{\mathrm{T}} + AP_{x_a x_b}B^{\mathrm{T}} + BP_{x_b x_a}A^{\mathrm{T}} + BP_{x_b x_b}B^{\mathrm{T}}.$$

13. Nonlinear transformations of probability distributions couple higher order moments into the first two moments.
14. The number of moments involved in evaluating the covariance of an Nth-order polynomial transformation of a variate grows as $2 \times N$.
15. The cumulative size of the data structures for the coefficients of a power series expansion of an n-vector to order N grows as

$$\frac{n(n^N - 1)}{(n - 1)},$$

where n is the vector dimension and N is the highest power of the vector finite power series expansion. The number of arithmetic operations required to evaluate the expansion grows as

$$\frac{Nn^{N+1}}{n - 1} + \frac{n(1 - n^N)}{(n - 1)^2}.$$

PROBLEMS

3.1 A deck of 52 playing cards is divided into four equal "suits" of 13 cards each, with each suit labeled with a heart (♡), diamond (◇), club (♣), or spade (♠). The 13 cards within each suit are labeled with an "A" (ace), 2, 3, 4, 5, 6, 7, 8, 9, 10, "J" (jack), "Q" (queen), or "K" (king). In a fair deal from a full deck, each card is equally likely to be drawn.

(a) In drawing a single card, what is the probability of drawing a spade?

(b) An ace?

(c) The ace of spades?

3.2 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.

3.3 Show that

$$\binom{n+1}{k+1} = \binom{n}{k+1} + \binom{n}{k},$$

for all $n > k$.

3.4 How many ways are there to divide a deck of 52 cards into four piles of 13 each?

3.5 If a hand of 13 cards are drawn from a deck of 52, what is the probability that exactly 3 cards are spades?

3.6 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.

3.7 A hand of 13 cards is dealt from a well-randomized deck.

(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.

3.8 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 $\rho_{n-1,\,n}$ between Y_{n-1} and Y_n.

3.9 The random variables X and Y are independent and uniformly distributed between 0 and 1 (rectangular distribution). Find the probability density function of $Z = |X - Y|$.

3.10 Two random variables x and y have the density function

$$p_{xy}(x, y) = \begin{cases} C(y - x + 1), & 0 \le y \le x \le 1, \\ 0, & \text{elsewhere,} \end{cases}$$

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 $y = \mathrm{E}\langle Y|x\rangle$.

3.11 The random variable X has the probability density function

$$f_X(x) = \begin{cases} 2x, & 0 \le x \le 1, \\ 0, & \text{elsewhere.} \end{cases}$$

Find the following:

(a) The cumulative function $F_X(x)$.

(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 $\sigma^2[X]$.

3.12 Can a probability distribution on a domain $\mathcal{D}$ with integrable functions $\mathcal{F}$ be defined in terms of a linear functional $\mathcal{P}$ such that

(a) $\mathcal{P}(f) \ge 0$ for all $f \in \mathcal{F}$ with nonnegative values, and

(b) $\mathcal{P}(1) = 1$, where the function "1" has value 1 everywhere in $\mathcal{D}$?

3.13 Derive and sketch the cumulative probability function for the Dirac δ distribution defined in Example 3.2.

3.14 For a univariate Gaussian probability density with mean μ and variance σ^2,

(a) What is the value of its *mode* (argument of probability density function at its maximum probability density)?

(b) What is the value of its *median* (argument of probability density function at which the cumulative probability equals $\frac{1}{2}$)?

(c) What is the LMSE estimate $\hat{x}_{\mathrm{LMSE}}$ of $x \in \mathcal{N}(\mu,\ \sigma^2)$?

3.15 Sketch the erf function, using an arrow to show where it has the value zero.

3.16 Write a formula for the cumulative probability function of the univariate Gaussian probability density with mean μ and variance σ^2, using the erf function.

3.17 The m-file `pYArctanaX.m` on the companion Wiley web site computes the function p_y given in Example 3.9. Write a script using MATLAB software calling `pYArctanaX(y,a)` to compute and plot the mean and variance of the resulting probability distribution as a function of the parameter a over the ranges $-\pi/2 \le y \le +\pi/2$ and $0.1 \le a \le 10$.

3.18 Use the m-file `pYArctanaX.m` with $a = 10$ (bimodal distribution) and MATLAB software to compute and plot the mean-squared estimation error as a function of the estimate $\hat{x}$ over the range $-\pi/2 \le \hat{x} \le +\pi/2$. Where does it achieve its minimum value?

REFERENCES

[1] P. Billingsley, *Probability and Measure, Anniversary Edition*, John Wiley & Sons, Inc., New York, 2012.

[2] C. M. Grinstead and J. L. Snell, *Introduction to Probability*, 2nd ed., American Mathematical Society, Providence, RI, 1997.

[3] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, McGraw-Hill, New York, 2002.

[4] D. Shafer and V. Vovk, "The Sources of Kolmogorov's *Grundbegriffe*," *Statistical Science*, Vol. 21, No. 1, pp. 70–98, 2006.

[5] A. N. Kolmogorov, *Bundbegriffe der Wahrscheinlichkeitsrechnung*, Springer, Berlin, 1933.

[6] O. Hölder, "Ueber einen Mittelwertsatz," *Nachrichten von der Königl, Gesellschaft der Wissenschaften und der Georg-Augusts-Universität zu Göttingen*, No. 2, pp. 38–47, Band (Vol.) 1889.

[7] D. Simon, *Optimal State Estimation: Kalman, H_∞, and Nonlinear Applications*, John Wiley & Sons, Inc., Hoboken, NJ, 2006.

[8] P. E. Greenwood and S. N. Nikulin, *A Guide to Chi-Squared Testing*, John Wiley & Sons, Inc., New York, 1996.

[9] F. C. Schweppe, "Evaluation of likelihood functions for Gaussian signals," *IEEE Transactions on Information Theory*, Vol. IT-11, pp. 61–70, 1965.

4

RANDOM PROCESSES

A completely satisfactory definition of random sequence is yet to be discovered.
—G. James and R. C. James, Mathematics Dictionary, Van Nostrand, Princeton NJ, 1959

4.1 CHAPTER FOCUS

Chapter 2 was about 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.

Chapter 3 was about probability distributions and their statistical parameters, how the parameters evolve under transformations of the variates, and properties of the parameters that do not depend on details of the underlying probability distributions.

In this chapter, some of the basic notions and mathematical models of statistical and deterministic mechanics are combined into a *stochastic system model*, which represents the evolution over time of key statistical parameters in systems with uncertain dynamics.

These stochastic system models are used to define random processes (RPs) in continuous time and in discrete time (also called *random sequences*). They represent the *state of knowledge* about a dynamic system—including its state of uncertainty. They represent *what we know* about a dynamic system, including a quantitative model for what we do not know.

Kalman Filtering: Theory and Practice Using MATLAB®, Fourth Edition.
Mohinder S. Grewal and Angus P. Andrews.
© 2015 John Wiley & Sons, Inc. Published 2015 by John Wiley & Sons, Inc.

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

4.1.1 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.

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 statistical properties such 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 5. The *orthogonality principle* is developed and explained with scalar examples. This principle will be used in Chapter 5 to derive the Kalman filter equations.

4.1.2 Topics Not Covered

The stochastic calculus for dynamic systems with white process noise is not defined, although its results are used. The interested reader is referred to books on the mathematics of stochastic differential equations (e.g., those by Allen [1], Arnold [2], Baras and Mirelli [3], Itô and McKean [4], Oksendal [5], Sobczyk [6], or Stratonovich [7]).

4.2 RANDOM VARIABLES, PROCESSES, AND SEQUENCES

4.2.1 Historical Background

4.2.1.1 Random Processes We have 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 for widows of professors at the University of Göttingen. Danish astronomer-turned-actuary Thorvald Nicolai Thiele (1838–1910) did some seminal work on modeling of RPs and sequences, and French mathematician Louis Bachelier (1870–1946) developed models for prices on the Paris Bourse, a 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.

Mathematicians and mathematical physicists have also been interested in modeling RPs in nature.

An early impetus for the development of a mathematical theory of stochastic systems was the 1828 publication of "A brief account of microscopical observations made on the particles contained in the pollen of plants and on the general existence of active molecules in organic and inorganic bodies" by the British botanist Robert Brown. In it, Brown described 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 has been studied extensively—both empirically and theoretically—by many eminent scientists (including Albert Einstein [8]) 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.

Mathematical models for RPs were derived in terms of what has come to be called *stochastic differential equations*. French scientist Paul Langevin[1] (1872–1946) modeled the velocity v of a particle in Brownian motion in terms of a differential equation [9] of the form

$$\frac{dv}{dt} = -\beta v + a(t), \tag{4.1}$$

where v is the velocity of a particle, β is a damping coefficient (due to the viscosity of the suspending medium), and $a(t)$ is called a *random force*. Equation 4.1 is now called the *Langevin equation.*

What Brown saw was a particle zig-zagging under the influence of velocity changes imparted by collisions with water molecules in an aqueous solution. The accelerations applied could be ascribed to van der Waals forces, which are sufficiently smooth that the velocity changes would not be instantaneous.

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

However, the random forcing function $a(t)$ of the Langevin equation (Equation 4.1) 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.

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*) for this model. Equivalent derivations were done by Russian mathematician Ruslan L. Stratonovich (1930–1997) [7] and others, and the stochastic calculus is now commonly used in modeling of RPs.

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 4.1 with $\beta = 0$, a process now called a *Wiener process.*

4.2.2 Definitions

A *random variable* (RV) X was defined in Chapter 3 in terms of a probability measure. For the purposes of this book, the values of RVs will almost always be n-dimensional real vectors.

An RP assigns a RV $X(t)$ to every time t on some interval. (When a variate appears in differential equation, the lowercase letter might be substituted, however.)

A **random sequence** (RS) assigns a RV X_k to every integer k in some range of integers. A random sequence may also be denoted by curly brackets as $\{x_k\}$, in which case the lowercase letter might be used.

In Kalman filtering, the statistical properties of interest for RPs and sequences include how their statistics and joint statistics are related across time or between components of vectors.

4.3 STATISTICAL PROPERTIES

4.3.1 Independent Identically Distributed (i.i.d.) Processes

For *identically distributed* (i.d.) processes and sequences, the RV distribution is identical for all values of time (t) or index (k). Figure 4.1(a) shows an example of

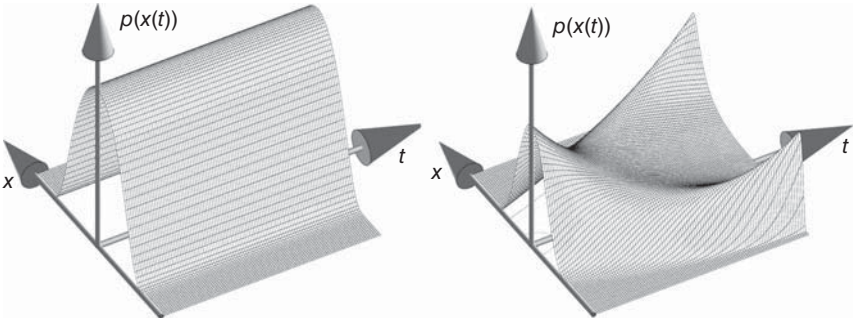


Figure 4.1 Probability densities of (a) identically distributed (i.d.) and (b) non-identically distributed processes.

a one-dimensional i.d. process, and Figure 4.1(b) shows an example of a non-i.d. process for which the shape of the probability density function evolves over time.

A process $X(t)$ is considered *time independent* if for any choice of distinct times $t_1, t_2, \ldots t_n$, the RVs $X(t_1), X(t_2), \ldots, X(t_n)$ are independent RVs. That is, their joint probability density is equal to the product of their individual probability densities:

$$p[x(t_1),\ x(t_2),\ \ldots,\ x(t_n)] = \prod_{i=1}^{n} p[x(t_i)]. \tag{4.2}$$

A similar definition for an independent identically distributed (i.i.d.) RS has k_i in place of t_i.

An *i.i.d. RP* is both i.d. and *independent*. Discrete and continuous i.i.d. processes play a significant role in Kalman filtering.

4.3.2 Process Means

For non-i.d. processes and sequences, the *mean* μ_x of an RP $X(t)$ or random sequence $\{X_k\}$ may be a function of time. For n-vector-valued RPs or RSs, the means are defined by the individual expected value taken at each time (continuous or discrete):

$$\mu_x(t) \stackrel{\text{def}}{=} \mathop{\mathrm{E}}_{x(t)\in X(t)} \langle x(t) \rangle \tag{4.3}$$

$$\mu_x(k) \stackrel{\text{def}}{=} \mathop{\mathrm{E}}_{x(k)\in X(k)} \langle x(k) \rangle, \tag{4.4}$$

respectively.

If the associated probability distributions $X(t)$ or $\{X_k\}$ can be defined by integrable probability density functions $p(\cdot)$, then the means can be defined in terms of

probability integrals as

$$\mu_x(t) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} x(t) p\ (x(t))\ dx(t), \tag{4.5}$$

$$\mu_x(k) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} x(k) p\ (x(k))\ dx(k), \tag{4.6}$$

which can be defined element-wise as

$$\mathop{\mathrm{E}}_{x(t)\in X(t)} \langle x_i(t)\rangle = \int_{-\infty}^{\infty} x_i(t)\ p(x_i(t))\ dx_i(t) \quad i = 1,\ 2,\ \ldots,n \tag{4.7}$$

$$\mathop{\mathrm{E}}_{x(k)\in -} \langle x_i(k)\rangle = \int_{-\infty}^{\infty} x_i(k)\ p(x_i(k))\ dx_i(k) \quad i = 1,\ 2,\ \ldots,n. \tag{4.8}$$

4.3.2.1 Zero-Mean Processes in Kalman Filtering Dynamic models for Kalman filtering separate zero-mean inputs from other inputs—usually labeled as (known) "control inputs" or (unknown) "slow variables." Any nonzero-mean RP would then be separated into its mean component and its (unknown) zero-mean component. If the mean of an RP were to change unpredictably and slowly over time, then its mean would be modeled as a separate "slow variable," not as part of a short-term RP. Models for such slow variables are an integral part of Kalman filtering.

Therefore, the fundamental RP models used in Kalman filtering will be *zero-mean process models*. Models for all other slowly varying parameters or variables can be built up from zero-mean RP models—which are usually i.i.d. processes, as well.

4.3.3 Time Correlation and Covariance

The *time correlation* of the n-vector-valued process $X(t)$ between any two times t_1 and t_2 is defined as the $n \times n$ matrix

$$\mathop{\mathrm{E}}_{\substack{x(t_1)\in X(t_1)\\ x(t_2)\in X(t_2)}} \langle x(t_1)x^T(t_2)\rangle = \mathop{\mathrm{E}}_{\substack{x(t_1)\in X(t_1)\\ x(t_2)\in X(t_2)}} \left\langle \begin{bmatrix} x_1(t_1)x_1(t_2) & \cdots & x_1(t_1)x_n(t_2) \\ \vdots & \ddots & \vdots \\ x_n(t_1)x_1(t_2) & \cdots & x_n(t_1)x_n(t_2) \end{bmatrix} \right\rangle \tag{4.9}$$

$$= \begin{bmatrix} \mathrm{E}\,\langle x_1(t_1)x_1(t_2)\rangle & \cdots & \mathrm{E}\,\langle x_1(t_1)x_n(t_2)\rangle \\ \vdots & \ddots & \vdots \\ \mathrm{E}\,\langle x_n(t_1)x_1(t_2)\rangle & \cdots & \mathrm{E}\,\langle x_n(t_1)x_n(t_2)\rangle \end{bmatrix}, \tag{4.10}$$

where, if the distributions can be defined in terms of probability density functions,

$$\mathrm{E}\,\langle x_i(t_1)x_j(t_2)\rangle = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x_i(t_1)x_j(t_2)p[x_i(t_1), x_j(t_2)]dx_i(t_1)dx_j(t_2). \tag{4.11}$$

4.3.3.1 Covariance Across Time If the RP involved is time independent but not zero-mean, time correlation can still be nonzero—even for an i.i.d. process. That is, if

$$p[x(t_1),\ x(t_2)] = p[x(t_1)] \times p[x(t_2)], \tag{4.12}$$

then for any i,j such that $1 \le i \le n$ and $1 \le j \le n$ and $t_1 \ne t_2$, the cross-correlation in time

$$\mathrm{E}\,\langle x_i(t_1)x_j(t_2)\rangle = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} x_i(t_1)x_j(t_2)p\,[x_i(t_1),\ x_j(t_2)]\,dx_i(t_1)dx_j(t_2) \tag{4.13}$$

$$= \left\{\int_{-\infty}^{+\infty} x_i(t_1)p\left[x_i\left(t_1\right)\right]dx_i(t_1)\right\}$$

$$\times\left\{\int_{-\infty}^{+\infty} x_j(t_2)p\left[x_j\left(t_2\right)\right]dx_j(t_2)\right\} \tag{4.14}$$

$$= \underset{x(t_1)\in X(t_1)}{\mathrm{E}}\langle x_i\rangle \times \underset{x(t_2)\in X(t_2)}{\mathrm{E}}\langle x_j\rangle, \tag{4.15}$$

which is the product of the means.

This issue can be avoided by defining the *covariance* across time by subtracting the means before taking expected values:

$$\underset{\substack{x(t_1)\in X(t_1)\\ x(t_2)\in X(t_2)}}{\mathrm{E}}\langle[x(t_1)-\mathrm{E}x(t_1)][x(t_2)-\mathrm{E}x(t_2)]^{\mathrm{T}}\rangle. \tag{4.16}$$

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

4.3.3.2 Cross-time-correlation and Covariance between RPs The time-cross-correlation matrix of two RPs $X(t)$, an n-vector, and $Y(t)$, an m-vector, is given by an $n \times m$ matrix

$$\mathrm{E}\langle x(t_1)y^{\mathrm{T}}(t_2)\rangle, \tag{4.17}$$

where

$$\mathrm{E}x_i(t_1)y_j(t_2) = \int_{-\infty}^{\infty}\int x_i(t_1)y_j(t_2)p[x_i(t_1),y_j(t_2)]dx_i(t_1)dy_j(t_2). \tag{4.18}$$

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

$$\mathrm{E}\langle[x(t_1)-\mathrm{E}x(t_1)][y(t_2)-\mathrm{E}y(t_2)]^{\mathrm{T}}\rangle. \tag{4.19}$$

4.3.4 Uncorrelated and Orthogonal Random Processes

4.3.4.1 Uncorrelated Random Processes A *random process* $X(t)$ is called *uncorrelated* if its *time covariance*

$$\mathrm{E}\,\langle [x(t_1) - \mathrm{E}\,\langle x(t_1)\rangle][x(t_2) - \mathrm{E}\,\langle x(t_2)\rangle]^{\mathrm{T}}\rangle = Q(t_1, t_2)\delta(t_1 - t_2), \tag{4.20}$$

where $\delta(t)$ is the Dirac delta "function," defined by

$$\int_a^b \delta(t)\, dt = \begin{cases} 1 & \text{if } a \le 0 \le b, \\ 0 & \text{otherwise.} \end{cases} \tag{4.21}$$

Similarly, a *random sequence* x_k is called *uncorrelated* if

$$\mathrm{E}\,\langle [x_k - \mathrm{E}\,\langle x_k\rangle][x_j - \mathrm{E}\,\langle x_j\rangle]^{\mathrm{T}}\rangle = Q(k, j)\Delta(k - j), \tag{4.22}$$

where $\Delta(\cdot)$ is the Kronecker delta function,[2] defined by

$$\Delta(k) = \begin{cases} 1 & \text{if } k = 0 \\ 0 & \text{otherwise.} \end{cases} \tag{4.23}$$

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:

$$\mathrm{E}\,\langle [x(t_1) - \mathrm{E}\,\langle x(t_1)\rangle][y(t_2) - \mathrm{E}\,\langle y(t_2)\rangle]^{\mathrm{T}}\rangle = 0. \tag{4.24}$$

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

4.3.4.2 Orthogonal Random Processes The processes $X(t)$ and $Y(t)$ are called *orthogonal* if their *correlation matrix* is identically zero:

$$\mathrm{E}\,\langle x(t_1)y^{\mathrm{T}}(t_2)\rangle = 0. \tag{4.25}$$

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

$$p[x(t_1), \ldots, x(t_n)] = \prod_{i=1}^{n} p[x(t_i)]. \tag{4.26}$$

Independence (all of the moments) also 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. 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 5).

[2]Named after the German mathematician Leopold Kronecker (1823–1891).

4.3.5 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:

$$p(x_1, x_2, \ldots, x_n, t_1, \ldots, t_n) = p(x_1, x_2, \ldots, x_n, t_1 + \varepsilon, t_2 + \varepsilon, \ldots, t_n + \varepsilon). \tag{4.27}$$

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

$$\mathrm{E}\,\langle x(t)\rangle = c \text{ (a constant)} \tag{4.28}$$

and

$$\mathrm{E}\,\langle x(t_1)x^{\mathrm{T}}(t_2)\rangle = Q(t_2 - t_1) = Q(\tau), \tag{4.29}$$

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.

4.3.6 Ergodic Random Processes

4.3.6.1 Historical Note 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.

Following Maxwell's hypothesis, an RP is considered **ergodic** 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 its ensemble averages.

4.3.7 Markov Processes and Sequences

An RP $X(t)$ is called a *Markov process*[3] if its future state distribution, conditioned on knowledge of its present state, is not improved by knowledge of previous

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

states:

$$p[x(t_i)|x(\tau);\tau < t_{i-1}] = p[x(t_i)|x(t_{i-1})], \tag{4.30}$$

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

Similarly, an RS x_k is called a *Markov sequence* if

$$p[x_i|x_k; k \leq i-1] = p[x_i|x_{i-1}]. \tag{4.31}$$

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

$$\dot{x}(t) = F(t)x(t) + G(t)w(t) \tag{4.32}$$

or

$$x_k = \Phi_{k-1}x_{k-1} + G_{k-1}w_{k-1}, \tag{4.33}$$

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

4.3.8 Gaussian Random Processes

An n-dimensional RP $X(t)$ is called *Gaussian* (or normal) if its probability density function is Gaussian, as given by the formulas of Example 3.1, with covariance matrix

$$P = \mathrm{E}\left\langle |x(t) - \mathrm{E}\left\langle x(t)\right\rangle [x(t) - \mathrm{E}\left\langle x(t)\right\rangle]^{\mathrm{T}}\right\rangle \tag{4.34}$$

for the RV $x \in X$.

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 RP results in another Gaussian RP.
4. All statistics of a Gaussian RP are completely determined by its first- and second-order statistics.

4.3.9 Simulating Multivariate Gaussian Processes

By using the Cholesky decomposition algorithm, Gaussian n-vector RPs with specified means and covariances can be simulated using scalar Gaussian pseudorandom number generators, such as randn in MATLAB software©.

Cholesky decomposition methods are discussed in Chapters 7 and 8, and in Appendix B (on the Wiley website). We show here how these methods can be used

to generate uncorrelated pseudorandom vector sequences with zero mean (or any specified mean) and a specified covariance Q.

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

$$s_i \in \mathcal{N}(0, 1) \text{ for all } i, \tag{4.35}$$

$$\mathrm{E}\,\langle s_i s_j \rangle = \begin{cases} 0 & \text{if} \quad i \neq j, \\ 1 & \text{if} \quad i = j. \end{cases} \tag{4.36}$$

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

$$u_k = [s_{nk+1},\ s_{nk+2},\ s_{nk+3},\ \dots,\ s_{n(k+1)}]^{\mathrm{T}}, \tag{4.37}$$

$$\mathrm{E}\,\langle u_k \rangle = 0, \tag{4.38}$$

$$\mathrm{E}\,\langle u_k u_k^{\mathrm{T}} \rangle = I_n. \tag{4.39}$$

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

$$CC^{\mathrm{T}} = Q \tag{4.40}$$

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

$$w_k = Cu_k. \tag{4.41}$$

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

$$\mathrm{E}\,\langle w_k \rangle = C\,\mathrm{E}\,\langle u_k \rangle \tag{4.42}$$

$$= 0 \tag{4.43}$$

(an n-vector of zeros) and covariance

$$\mathrm{E}\,\langle w_k w_k^{\mathrm{T}} \rangle = \mathrm{E}\,\langle Cu_k (Cu_k)^{\mathrm{T}} \rangle \tag{4.44}$$

$$= CI_n C^{\mathrm{T}} \tag{4.45}$$

$$= Q. \tag{4.46}$$

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.

4.3.10 Power Spectral Densities

Let $X(t)$ be a zero-mean scalar stationary RP with autocorrelation function

$$\psi_x(\tau) = \underset{t}{\mathrm{E}}\langle x(t)x(t+\tau)\rangle. \tag{4.47}$$

The PSD is defined as the Fourier transform $\Psi_x(\omega)$ of $\psi_x(\tau)$,

$$\Psi_x(\omega) = \int_{-\infty}^{\infty} \psi_x(\tau) e^{-j\omega\tau}\, d\tau, \tag{4.48}$$

where the inverse transform as

$$\psi_x(\tau) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \Psi_x(\omega) e^{j\omega\tau}\, d\omega. \tag{4.49}$$

The following are useful properties of autocorrelation functions:

1. Autocorrelation functions are symmetrical ("even" functions).
2. An autocorrelation 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 all valid autocorrelation functions.

Setting $\tau = 0$ in Equation 4.49 gives

$$\underset{t}{\mathrm{E}}\langle x^2(t)\rangle = \psi_x(0) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \Psi_x(\omega)\, d\omega. \tag{4.50}$$

Because of property 1 of the autocorrelation function,

$$\Psi_x(\omega) = \Psi_x(-\omega); \tag{4.51}$$

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

Example 4.1 (PSD for an Exponentially Correlated Process) The exponentially correlated process diagrammed in Figure 4.2(a) has an autocovariance function of the general form

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

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

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

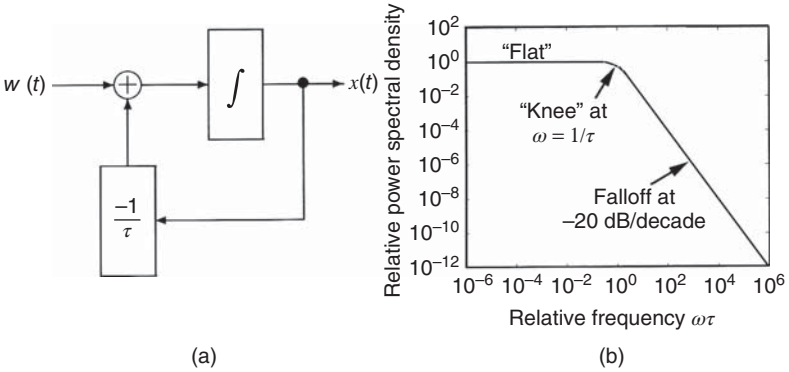


Figure 4.2 Exponentially correlated random process. (a) Block diagram and (b) PSD.

the shape of which is illustrated in Figure 4.2(b) as a log–log plot. White noise passed through a low pass resistor–capacitor (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 $\omega = 1/\tau = 1/(R \times C)$.

Example 4.2 (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, 2.6, and 2.7 and will be used again in Chapters 5 and 7.

The transfer function of the dynamic system is

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

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

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

The general form of the autocorrelation is

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

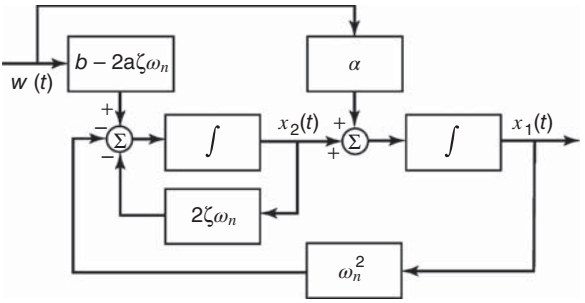


Figure 4.3 Diagram of a second-order Markov process

In practice, σ^2, θ, ζ, and w_n are chosen to fit empirical data (see Problem 4.4). The PSD corresponding to the $\psi_x(\tau)$ will have the form

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

(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 4.3.

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

$$\mathop{\mathrm{E}}_{t}\langle x^2(t)\rangle = \lim_{T\to\infty}\int_{-T}^{\mathrm{T}} x^2(t)\ dt \tag{4.52}$$

$$= \frac{1}{2\pi}\int_{-\infty}^{\infty}\Psi_x(\omega)\ d\omega \tag{4.53}$$

$$= \sigma^2. \tag{4.54}$$

The *cross power spectral density* between an RP $X(t)$ and an RP $Y(t)$ is given by the formula

$$\Psi_{xy}(\omega) = \int_{-\infty}^{\infty}\psi_{xy}(\tau)e^{-j\omega\tau}\ d\tau. \tag{4.55}$$

4.4 LINEAR RANDOM PROCESS MODELS

Linear system models of the type illustrated in Figure 4.4 are defined by the equation

$$y(t) = \int_{-\infty}^{\infty} x(\tau)h(t,\tau)\ d\tau, \tag{4.56}$$

where $X(t)$ is input and $h(t, \tau)$ is the linear system *weighting function* (see Figure 4.4). If the system is time invariant (i.e., h does not depend on t), then Equation 4.56 can

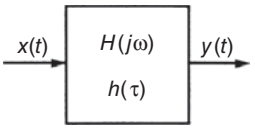


Figure 4.4 Block diagram representation of a linear system.

be written as

$$y(t) = \int_0^\infty h(\tau)x(t-\tau)\,d\tau. \tag{4.57}$$

This type of integral is called a *convolution integral,* and $h(\tau)$ is called its *kernel function.* Manipulation of Equation 4.57 leads to relationships between autocorrelation functions of $X(t)$ and $Y(t)$,

$$\psi_y(\tau) = \int_0^\infty d\tau_1 h(\tau_1) \int_0^\infty d\tau_2 h(\tau_2)\psi_x(\tau + \tau_1 - \tau_2)\,\text{(autocorrlation,)} \tag{4.58}$$

$$\psi_{xy}(\tau) = \int_0^\infty h(\tau_1)\psi_x(\tau - \tau_1)\,d\tau_1\,\text{(cross-correlation,)} \tag{4.59}$$

and spectrum relationships

$$\Psi_{xy}(\omega) = H(j\omega)\Psi_x(\omega)\,\text{(cross-spectrum)}, \tag{4.60}$$

$$\Psi_y(\omega) = |H(j\omega)|^2\Psi_x(\omega)\,\text{(PSD)}, \tag{4.61}$$

where H is the *system transfer function* (also shown in Figure 4.4), defined by the Laplace transform of $h(\tau)$ as

$$\int_0^\infty h(\tau)e^{s\tau}\,d\tau = H(s) = H(j\omega), \tag{4.62}$$

where $s = j\omega$ and $j = \sqrt{-1}$.

4.4.1 Stochastic Differential Equations for RPs

4.4.1.1 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 [10] and Itô [11].

4.4.1.2 Linear Stochastic Differential Equations A linear stochastic differential equation as a model of an RP with initial conditions has the general form

$$\dot{x}(t) = F(t)x(t) + G(t)w(t) + C(t)u(t), \tag{4.63}$$

$$z(t) = H(t)x(t) + v(t) + D(t)u(t), \tag{4.64}$$

where the variables are defined as

$x(t)$ is an $n \times 1$ state vector,
$z(t)$ is an $l \times 1$ measurement vector,
$u(t)$ is an $r \times 1$ deterministic input vector,
$F(t)$ is an $n \times n$ time-varying dynamic coefficient matrix,
$C(t)$ is an $n \times r$ time-varying input coupling matrix,
$H(t)$ is an $l \times n$ time-varying measurement sensitivity matrix,
$D(t)$ is an $l \times r$ time-varying output coupling matrix,
$G(t)$ is an $n \times r$ time-varying process noise coupling matrix,
$w(t)$ is an $r \times 1$ zero-mean uncorrelated "plant noise" process,
$v(t)$ is an $l \times 1$ zero-mean uncorrelated "measurement noise" process,

and the expected values will be

$$\begin{aligned}
\mathrm{E}\,\langle w(t)\rangle &= 0,\\
\mathrm{E}\,\langle v(t)\rangle &= 0,\\
\mathrm{E}\,\langle w(t_1)w^{\mathrm{T}}(t_2)\rangle &= Q(t_1)\delta(t_2 - t_1),\\
\mathrm{E}\,\langle v(t_1)v^{\mathrm{T}}(t_2)\rangle &= R(t_1)\delta(t_2 - t_1).\\
\mathrm{E}\,\langle w(t_1)v^{\mathrm{T}}(t_2)\rangle &= M(t_1)\delta(t_2 - t_1)
\end{aligned}$$

$$\delta(t) = \begin{cases} 1, & t = 0,\\ 0, & t \neq 0. \end{cases} \qquad \text{(Dirac delta function)}$$

The symbols Q, R, and M represent $r \times r$, $l \times l$, and $r \times l$ matrices, respectively, and δ represents the Dirac delta "function" (a measure). The values over time of the variate $X(t)$ 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 4.3 (Exponentially Correlated Process) Continuing with Example 4.1, let the RP $X(t)$ be a zero-mean stationary normal RP having autocorrelation

$$\psi_x(\tau) = \sigma^2 e^{-\alpha|\tau|}. \tag{4.65}$$

The corresponding PSD is

$$\Psi_x(\omega) = \frac{2\sigma^2\alpha}{\omega^2 + \alpha^2}. \tag{4.66}$$

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 4.61, 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

$$H(s) = \frac{\sqrt{2\alpha}\sigma}{s + \alpha}, \tag{4.67}$$

which can be represented as

$$\frac{x(s)}{w(s)} = \frac{\sqrt{2\alpha}\sigma}{s + \alpha}, \tag{4.68}$$

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 called the *correlation time* of the process.

The block diagram representation of the process in Example 4.3 is shown in Table 4.1. This is called a *shaping filter*. Some other examples of differential equation models are also given in Table 4.1.

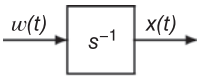
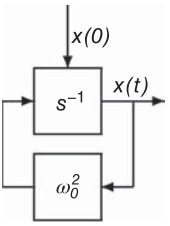
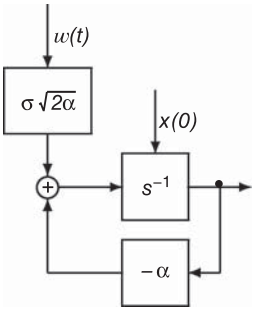
4.4.2 Discrete-Time Models for Random Sequences (RS)

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

$$\begin{aligned} x_k &= \Phi_{k-1}x_{k-1} + G_{k-1}w_{k-1} + \Gamma_{k-1}u_{k-1}, \\ z_k &= H_k x_k + \upsilon_k + D_k u_k. \end{aligned} \tag{4.69}$$

This is the complete model with deterministic inputs u_k as discussed in Chapter 2, Equations 2.38 and 2.39, and RS noise w_k and υ_k as described in Chapter 5,

TABLE 4.1 System Models of Random Processes

Random Process	Autocorrelation Function and Power Spectral Density	Shaping Filter	State-Space Model
White noise	$\psi_x(\tau) = \sigma^2\delta^2(\tau)$ $\Psi_x(\omega) = \sigma^2$	None	Plant noise $w(t)$
Random walk	$\psi_x(\tau) = \mathrm{E}\langle x(t_1)x(t_2)\rangle$ $= \sigma^2 t_1$ if $t_1 < t_2$ $\Psi_x(\omega) \propto \sigma^2/\omega^2$		$\dot{x} = w(t)$ $\sigma_x^2(0) = 0$
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	$\psi_x(\tau) = \sigma^2\cos(\omega_0\tau)$ $\Psi_x(\omega) = \pi\sigma^2\delta(\omega - \omega_0)$		$\dot{x} = \begin{bmatrix} 0 & 1 \\ -\omega_0^2 & 0 \end{bmatrix} x$ $P(0) = \begin{bmatrix} \sigma^2 & 0 \\ 0 & 0 \end{bmatrix}$
Exponentially correlated	$\psi_x(\tau) = \sigma^2 e^{-\alpha\lvert\tau\rvert}$ $\Psi_x(\omega) = \frac{2\sigma^2\alpha}{\omega^2+\alpha^2}$ $1/\alpha$ = correlation time		$\dot{x} = -\alpha x + \sigma\sqrt{2\alpha}w(t)$ $\sigma_x^2(0) = \sigma^2$

where

x_k is an $n \times 1$ state vector,
z_k is an $\ell \times 1$ measurement vector,
u_k is an $r \times 1$ deterministic input vector,
Φ_{k-1} is an $n \times n$ time-varying matrix,
G_{k-1} is an $n \times r$ time-varying matrix,
H_k is an $\ell \times n$time-varying matrix,
D_k is an $\ell \times r$ time-varying matrix,
Γ_{k-1} is an $n \times r$ time-varying matrix,

and the expected values

$$\begin{aligned}
\mathrm{E}\,\langle w_k \rangle &= 0 \\
\mathrm{E}\,\langle v_k \rangle &= 0 \\
\mathrm{E}\,\langle w_{k_1} w_{k_2}^{\mathrm{T}} \rangle &= Q_{k_1} \Delta(k_2 - k_1) \\
\mathrm{E}\,\langle v_{k_1} v_{k_2}^{\mathrm{T}} \rangle &= R_{k_1} \Delta(k_2 - k_1) \\
\mathrm{E}\,\langle w_{k_1} v_{k_2}^{\mathrm{T}} \rangle &= M_{k_1} \Delta(k_2 - k_1).
\end{aligned}$$

Example 4.4 (Exponentially Correlated Sequence) 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

$$x_k = \Phi x_{k-1} + G w_{k-1}, \quad z_k = x_k. \tag{4.70}$$

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 4.19 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

$$\Phi = e^{-\alpha}. \tag{4.71}$$

Next, square the state variable defined by Equation 4.70 and take its expected value:

$$\mathrm{E}\,\langle x_k^2 \rangle = \Phi^2 \mathrm{E}\,\langle x_{k-1} x_{k-1} \rangle + G^2 \mathrm{E}\,\langle w_{k-1} w_{k-1} \rangle, \tag{4.72}$$

$$\sigma^2 = \sigma^2 \Phi^2 + G^2, \tag{4.73}$$

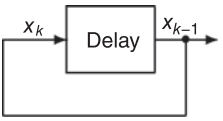
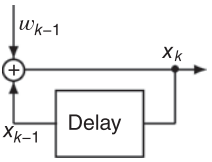
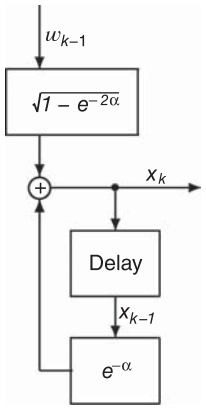
because the variance $\mathrm{E}\,\langle w_{k-1}^2 \rangle = 1$ and the parameter $G = \sigma\sqrt{1 - e^{-2\alpha}}$.

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 $\mathrm{E}\,\langle w_{k_1}\, w_{k_2} \rangle = \Delta(k_2 - k_1)$.

TABLE 4.2 Stochastic System Models for Discrete Random Sequences

Process Type	Autocorrelation Function	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	$\rightarrow +\infty$		$x_k = x_{k-1} + w_{k-1}$ $\sigma_x^2(0) = 0$
Exponentially correlated	$\psi_x(k_2 - k_1) = \sigma^2 e^{-\alpha\lvert k_2 - k_1\rvert}$		$x_k = e^{-\alpha} x_{k-1} + \sigma\sqrt{1 - e^{-2\alpha}}\, w_{k-1}$ $\sigma_x^2(0) = \sigma^2$

The dynamic process model derived in Example 4.4 is called a shaping filter. Block diagrams of this and other shaping filters are given in Table 4.2, along with their difference equation models.

4.4.3 Autoregressive Processes and Linear Predictive Models

A *linear predictive model* for a signal is a representation in the form

$$\acute{x}_{k+1} = \sum_{i=1}^{n} a_i \acute{x}_{k-i+1} + \acute{u}_k, \tag{4.74}$$

where $\acute{x}_k$ 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 sequence. Note that this formula for an autoregressive process can be rewritten in state-transition matrix (STM) form as

$$\begin{bmatrix} \acute{x}_{k+1} \\ \acute{x}_k \\ \acute{x}_{k-1} \\ \vdots \\ \acute{x}_{k-n+2} \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & \cdots & a_{n-1} & a_n \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \begin{bmatrix} \acute{x}_k \\ \acute{x}_{k-1} \\ \acute{x}_{k-2} \\ \vdots \\ \acute{x}_{k-n+1} \end{bmatrix} + \begin{bmatrix} \acute{x} \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \tag{4.75}$$

$$x_{k+1} = \Phi x_k + u_k, \tag{4.76}$$

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 u_k will be filled with zeros, except for the term $Q_{11} = \mathrm{E}\,\langle \acute{u}_k^2 \rangle$.

4.5 SHAPING FILTERS (SF) AND STATE AUGMENTATION

Shaping filters have a long and interesting history. In the early years of "wireless telegraphy" (radio) transmission, a spark provided a white-noise source which could be input to an LC "tank circuit" (a shaping filter), the output of which was the radio frequency (RF) transmission frequency band of interest.

The focus of this section is on the use of shaping filters to model nonwhite-noise models for stationary RPs, using white-noise processes as inputs. 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) [12] and by Lotfi Asker Zadeh and John Ralph Ragazzini (1912–1988) [13]. 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.

4.5.1 Correlated Process Noise Models

4.5.1.1 Shaping Filters for Dynamic Disturbance Noise Let a system model be given by

$$\dot{x}(t) = F(t)x(t) + G(t)w_1(t), \quad z(t) = H(t)x(t) + v(t), \tag{4.77}$$

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:[4]

$$\dot{x}_{\mathrm{SF}}(t) = F_{\mathrm{SF}}(t)x_{\mathrm{SF}}(t) + G_{\mathrm{SF}}(t)w_2(t) \tag{4.78}$$

$$w_1(t) = H_{\mathrm{SF}}(t)x_{\mathrm{SF}}(t), \tag{4.79}$$

where SF denotes the shaping filter and $w_2(t)$ is zero-mean white Gaussian noise. Now define a new augmented state vector

$$X(t) = [x(t)x_{\mathrm{SF}}(t)]^{\mathrm{T}}. \tag{4.80}$$

Equations 4.77 and 4.79 can be combined into the matrix form

$$\begin{bmatrix} \dot{x}(t) \\ \dot{x}_{\mathrm{SF}}(t) \end{bmatrix} = \begin{bmatrix} F(t) & G(t)H_{\mathrm{SF}}(t) \\ 0 & F_{\mathrm{SF}}(t) \end{bmatrix} \begin{bmatrix} x(t) \\ x_{\mathrm{SF}}(t) \end{bmatrix} + \begin{bmatrix} 0 \\ G_{\mathrm{SF}}(t) \end{bmatrix} w_2(t), \tag{4.81}$$

$$\dot{X}(t) = F_T(t)X(t) + G_T(t)w_2(t), \tag{4.82}$$

and the output equation can be expressed in compatible format as

$$z(t) = [H(t)\ 0] \begin{bmatrix} x(t) \\ x_{\mathrm{SF}}(t) \end{bmatrix} + v(t) \tag{4.83}$$

$$= H_T(t)X(t) + v(t). \tag{4.84}$$

This total system given by Equations 4.82 and 4.84 is a linear differential equation model driven by white Gaussian noise. (See Figure 4.5 for a non-white noise model.)

4.5.2 Correlated Measurement Noise Models

4.5.2.1 Shaping Filters for Measurement Noise A similar development is feasible for the case of time-correlated measurement noise $v_1(t)$:

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

$$z(t) = H(t)x(t) + v_1(t). \tag{4.85}$$

In this case, let $v_2(t)$ be zero-mean white Gaussian noise and let the measurement noise $v_1(t)$ be modeled by

$$\dot{x}_{\mathrm{SF}}(t) = F_{\mathrm{SF}}(t)x_{\mathrm{SF}}(t) + G_{\mathrm{SF}}(t)v_2(t),$$

$$v_1(t) = H_{\mathrm{SF}}(t)x_{\mathrm{SF}}(t). \tag{4.86}$$

[4]See Example 4.2 for WSS processes.

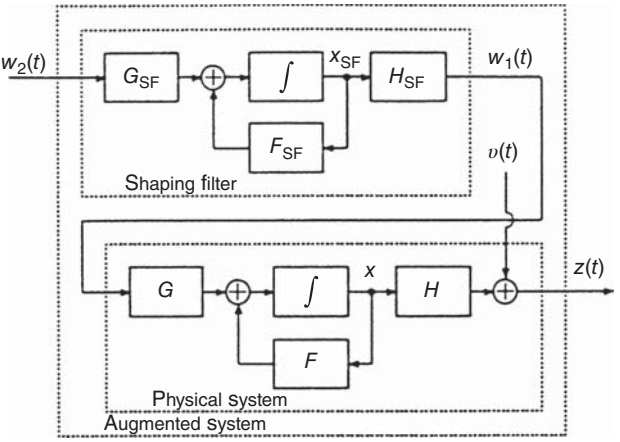


Figure 4.5 Shaping filter model for nonwhite noise.

The total augmented system is given by

$$\begin{bmatrix} \dot{x}(t) \\ \dot{x}_{SF}(t) \end{bmatrix} = \begin{bmatrix} F(t) & 0 \\ 0 & F_{SF}(t) \end{bmatrix} \begin{bmatrix} x(t) \\ x_{SF}(t) \end{bmatrix} + \begin{bmatrix} G(t) & 0 \\ 0 & G_{SF}(t) \end{bmatrix} \begin{bmatrix} w(t) \\ v_2(t) \end{bmatrix},$$

$$z(t) = [H(t)\ H_{SF}(t)] \begin{bmatrix} x(t) \\ x_{SF}(t) \end{bmatrix}. \tag{4.87}$$

This is in the form of a linear system model driven by white Gaussian noise and output equation with no input noise.

These systems can be specialized to the WSS process for continuous and discrete cases as by shaping filters shown in Tables 4.1 and 4.2.

Example 4.5 (Accelerometer Error 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:

$$\Delta_{\beta_m} = \beta_m \otimes \zeta + b_A + h_A \beta_m + \beta_m^2 (FI1 - FX1) + \delta\beta,$$

$$h_A = \begin{bmatrix} S_1 & \delta_{12} & \delta_{13} \\ 0 & S_2 & \delta_{23} \\ 0 & 0 & S_3 \end{bmatrix}$$

$$\beta_m^2 = \begin{bmatrix} \beta_m^2 & 0 & 0 \\ 0 & \beta_2^2 & 0 \\ 0 & 0 & \beta_3^2 \end{bmatrix}$$

where

Δ_{β_m} is the midpoint acceleration error,
$\otimes$ is the cross product for 3-vectors,
ζ is a 3×1 vector representing attitude alignment errors between "platform" axes and computational axes,
b_A is a 3×1 vector of unknown accelerometer biases, normalized to the magnitude of gravity,
S_i are unknown accelerometer scale factor errors ($i = 1, 2, 3$),
δ_{ij} are unknown accelerometer axes nonorthogonalities,
$\delta\beta$ are other error terms, some of which are observable; for reason of practicality in our example, they are not estimated, only compensated with factory-calibrated values,
$FI1$ is a 3×1 unknown acceleration-squared nonlinearity for acceleration along the accelerometer input axis,
$FX1$ is a 3×1 unknown acceleration-squared nonlinearity for acceleration normal to the accelerometer input axis, and
β_m is a 3×1vector$(\beta_1, \beta_2, \beta_3)^{\mathrm{T}}$ of midpoint components of acceleration in platform coordinates.

The 12×1 accelerometer state vector x^A is composed of the subvectors and scalars

$$(x^A) = \left[\underbrace{b_A^{\mathrm{T}}}_{1\times3} \quad S_1 \quad \delta_{12} \quad S_2 \quad \delta_{13} \quad \delta_{23} \quad S_3 \quad \underbrace{(FX1 - FI1)^{\mathrm{T}}}_{1\times3} \right]^{\mathrm{T}}.$$

The 12 unknown parameters can be modeled as random walks (see Table 4.1) for the parameter identification problem to be discussed in Chapter 10.

An even larger sensor model is described in the next example.

Example 4.6 (Gyroscope Error Model) A 48-state gyroscope drift error model is given as follows:

$$\varepsilon = b_g + h_g\omega + U_g\beta + K_g\beta^1 + \mathrm{diag}|\omega|T_g + b_{gt}t + U_{gt}t\beta,$$

$$h_g = \begin{bmatrix} S_{g1} & \Delta_{12} & \Delta_{13} \\ \Delta_{21} & S_{g2} & \Delta_{23} \\ \Delta_{31} & \Delta_{32} & S_{g3} \end{bmatrix}$$

$$U_g = \begin{bmatrix} d_{I1} & d_{01} & d_{S1} \\ d_{S2} & d_{I2} & d_{02} \\ d_{03} & d_{S3} & d_{I3} \end{bmatrix}$$

$$K_g = \begin{bmatrix} k_{II1} & k_{001} & k_{SS1} & I_{I01} & k_{IS1} & k_{S01} \\ k_{SS2} & k_{II2} & k_{002} & k_{IS2} & k_{S02} & k_{I02} \\ k_{003} & k_{SS3} & k_{II3} & k_{S03} & k_{I03} & k_{IS3} \end{bmatrix}$$

$$\beta^1 = \begin{bmatrix} \beta_1^2 & \beta_2^2 & \beta_3^2 & \beta_1\beta_2 & \beta_1\beta_3 & \beta_2\beta_3 \end{bmatrix}^{\mathrm{T}}$$

$$x^g(t) = \begin{bmatrix} 1\times 3 & 1\times 9 & 1\times 9 & 1\times 15 & 1\times 3 & 1\times 3 & 1\times 3 \\ b_g^{\mathrm{T}} & h_g^{1T} & U_g^{1\mathrm{T}} & K_g^{1\mathrm{T}} & T_g^{\mathrm{T}} & b_{gt}^{\mathrm{T}} & U_{gt}^{1\mathrm{T}} \end{bmatrix}^{\mathrm{T}},$$

where

x^g is the 48-state gyroscope subsystem state vector of drifting error parameters,

b_g is a 3×1 vector of unknown gyroscope fixed drift parameters,

h_g is a 3×1 matrix containing unknown scale factor (S_{gi}) and linear axes alignment errors (Δ_{ij}) as components ($i, j = 1, 2, 3$),

T_g is a 3×1 vector of unknown nonlinear gyroscope torquer scale factor errors, with elements δS_{gi},

$diag\ |\omega|$ is 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 is a 3×3 matrix of unknown gyroscope mass unbalance parameters ($d_{k,j}$), indices I, 0, and S denoting input, output, and spin axes, respectively, for each gyroscope 1, 2 and 3,

K_g is a 3×6 matrix of unknown gyroscope compliance (g-squared) errors k_{kji},

b_{gt} is a 3×1 vector of unknown gyroscope fixed drift trend parameters,

U_{gt} is a 3×6 matrix of unknown gyroscope mass unbalance trend parameters, and

β is a 3×1 vector of vertical direction cosines (normalized gravity) $(\beta_1, \beta_2, \beta_3)^{\mathrm{T}}$

β^1 is a 6×1 vector.

The 48 unknown parameters are modeled as random walks and random ramps (see Table 4.1) for the parameter identification problem to be discussed in Chapter 10 (GNSS/INS integration).

4.6 MEAN AND COVARIANCE PROPAGATION

The time propagation of the means and covariances of linear stochastic systems is fundamental to Kalman filtering.

This section combines the fundamental equations for solutions of nonhomogeneous linear differential equations in Chapter 2 and equations for linear transformations of means and covariances derived in Chapter 3 to obtain solutions to generic linear stochastic differential equations of the sort

$$\frac{d}{dt}x(t) = F(t)\ x(t) + G(t)\ w(t),$$

where $w(t)$ is a zero-mean white-noise process.

The following subsections address these issues for linear stochastic systems in continuous time and in discrete time.

4.6.1 Propagating the Mean

4.6.1.1 General Solution If the dynamics of an n-dimensional state vector x can be modeled by a nonhomogeneous linear differential equation with disturbance $u(t)$ of the sort

$$\frac{d}{dt}x(t) = F(t)x(t) + G(t)w(t), \tag{4.88}$$

then the general solution is given by Equation 2.26 in terms of its initial value at $t = t_0$ as

$$x(t) = \Phi(t, t_0)\ x(t_0) + \int_{t+0}^{t} \Phi(t, s)\ G(s)\ w(s)\ ds \tag{4.89}$$

$$\Phi\,(t, s) \stackrel{\text{def}}{=} \exp\ \left[\int_{s}^{t} F\,(\tau)\ d\tau\right]. \tag{4.90}$$

Consequently, for $x \in X$, a random variate, the mean can be propagated as

$$\mu_x(t) \stackrel{\text{def}}{=} \underset{x\in X}{\mathrm{E}}\langle x(t)\rangle \tag{4.91}$$

$$= \underset{x}{\mathrm{E}}\left\langle \Phi\left(t, t_0\right)\ x(t) + \int_{t+0}^{t} \Phi(t, s)\ G(s)\ w(s)\ ds \right\rangle \tag{4.92}$$

$$= \Phi(t, t_0)\ \underset{x}{\mathrm{E}}\,\langle x(t_0)\rangle + \int_{t+0}^{t} \Phi(t, s)\ G(s)\ w(s)\ ds \tag{4.93}$$

$$= \Phi(t, t_0)\ \mu_x(t_0) + \int_{t+0}^{t} \Phi(t, s)\ G(s)\ w(s)\ ds. \tag{4.94}$$

Generic Linear Stochastic Differential Equation Solution If the disturbance function $u(t) = w(t)$, a *white-noise process* with mean $\mathrm{E}_w\langle w(t)\rangle = 0$, the generic linear differential equation becomes a generic linear *stochastic* differential equation. In that case, the expected solution for the mean of x would be

$$\mu_x(t) = \Phi(t, t_0)\ \underset{x}{\mathrm{E}}\,\langle x(t_0)\rangle + \int_{t+0}^{t} \Phi(t, s)\ G(s)\ \underset{w}{\mathrm{E}}\,\langle w(s)\rangle\ ds \tag{4.95}$$

$$= \Phi(t, t_0)\ \mu_x(t_0) + \int_{t+0}^{t} \Phi(t, s)\ G(s)\ (0)\ ds \tag{4.96}$$

$$= \Phi(t, t_0)\ \mu_x(t_0), \tag{4.97}$$

$$= \exp\ \left[\int_{t_0}^{t} F\,(\tau)\ d\tau\right] \mu_x(t_0), \tag{4.98}$$

the general solution for the mean for the generic linear stochastic differential equation model.

4.6.1.2 Differential Equation Model for the Mean Equation 4.98 can be differentiated with respect to t to obtain a differential equation,

$$\frac{d}{dt}\mu_x(t) = \frac{d}{dt}\exp\left[\int_{t_0}^{t} F(\tau)\ d\tau\right]\mu_x(t_0) \tag{4.99}$$

$$= F(t)\ \exp\left[\int_{t_0}^{t} F(\tau)\ d\tau\right]\mu_x(t_0) \tag{4.100}$$

$$= F(t)\ \mu_x(t) \tag{4.101}$$

for propagating the mean in continuous time.

4.6.1.3 Discrete-Time Model for the Mean Equation 4.97 is also the solution in discrete time if we set $t_0 = t_{k-1}$ and $t = t_k$, in which case

$$\mu_{x,k} \overset{\text{def}}{=} \mathop{\mathrm{E}}_{x}\langle x_k\rangle \tag{4.102}$$

$$= \Phi_{k-1}\mu_{x,k-1} \tag{4.103}$$

$$\Phi_{k-1} \overset{\text{def}}{=} \Phi(t_k, t_{k-1}) \tag{4.104}$$

$$= \exp\left[\int_{t_{k-1}}^{t_k} F(s)\ ds\right]. \tag{4.105}$$

4.6.2 Propagating the Covariance

4.6.2.1 General Solution for the Stochastic System The covariance matrix

$$P_{xx}(t) \overset{\text{def}}{=} \mathop{\mathrm{E}}_{x}\langle[x(t) - \mu_x(t)][x(t) - \mu_x(t)]^{\mathrm{T}}\rangle, \tag{4.106}$$

where $x(t)$ is given by Equation 4.89 with $u(t) = w(t)$, $\mu_x(t)$ is given by Equation 4.97, and their difference

$$x(t) - \mu_x(t) = \Phi(t, t_0)\ [x(t_0) - \mu_x(t_0)] + \int_{t_0}^{t} \Phi(t, s)\ G(s)\ w(s)\ ds \tag{4.107}$$

$$\Phi(t, s) \overset{\text{def}}{=} \exp\left[\int_{s}^{t} F(\tau)\ d\tau\right]. \tag{4.108}$$

As a consequence, the covariance matrix

$$P_{xx}(t) = \mathrm{E}_{x,w}\left\langle\left[\Phi\left(t, t_0\right)\ [x(t_0) - \mu_x(t_0)] + \int_{t_0}^{t} \Phi(t, s)\ G(s)\ w(s)\ ds\right]\right.$$

$$\left.\times\left[\Phi\left(t, t_0\right)\ [x(t_0) - \mu_x(t_0)] + \int_{t_0}^{t} \Phi(t, s)\ G(s)\ w(s)\ ds\right]^{\mathrm{T}}\right\rangle \tag{4.109}$$

$$= \underset{x}{\mathrm{E}} \langle [\Phi(t,t_0)\ [x(t_0) - \mu_x(t_0)]][\Phi(t,t_0)\ [x(t_0) - \mu_x(t_0)]]^{\mathrm{T}} \rangle$$
$$+ \underset{w}{\mathrm{E}} \left\langle \left[\int_{t_0}^{t} \Phi\left(t, s_1\right)\ G(s_1)\ w(s_1)\ ds_1 \right] \right.$$
$$\left. \times \left[\int_{t_0}^{t} \Phi\left(t, s_2\right)\ G(s_2)\ w(s_2)\ ds_2 \right]^{\mathrm{T}} \right\rangle \tag{4.110}$$
$$= \Phi(t,t_0) P_{xx}(t_0) \Phi^{\mathrm{T}}(t,t_0)$$
$$+ \int_{t_0}^{t} \Phi(t,s)\ G(s) \underset{w}{\mathrm{E}} \langle w(s)\ w^{\mathrm{T}}(s) \rangle G^{\mathrm{T}}(s)\ \Phi^{\mathrm{T}}(t,s)\ ds \tag{4.111}$$
$$= \Phi(t,t_0) P_{xx}(t_0) \Phi^{\mathrm{T}}(t,t_0) + \int_{t_0}^{t} \Phi(t,s)\ G(s)\ Q_t(s)\ G^{\mathrm{T}}(s)\ \Phi^{\mathrm{T}}(t,s)\ ds, \tag{4.112}$$

where we have used the notation Q_t to denote the dynamic disturbance noise covariance in continuous time, which has units of squared deviation per unit of time. This is to distinguish it from the covariance of dynamic disturbance noise in discrete time, which has units of squared deviation.

In the sequence of equations from Equation 4.109 to Equation 4.112,

(a) The transition between Equations 4.109 and 4.110 used the fact that

$$\underset{w}{\mathrm{E}} \langle w(s) \rangle = 0.$$

(b) The transition between Equations 4.110 and 4.111 used the fact that

$$\underset{w}{\mathrm{E}} \langle w(s_1) w^{\mathrm{T}}(s_2) \rangle = 0$$

unless $s_1 = s_2$, which collapses the double integral down to a single integral.

(c) The transition between Equations 4.111 and 4.112 used the fact that

$$\underset{w}{\mathrm{E}} \langle w(s) w^{\mathrm{T}}(s) \rangle = Q_t(s).$$

4.6.2.2 Differential Equation Model If one writes Equation 4.112 in the alternative form

$$P_{xx}(t) = \exp\ \left[\int_{t_0}^{t} F(\tau)\ d\tau \right]\ P_{xx}(t_0)\ \exp\ \left[\int_{t_0}^{t} F(\tau)\ d\tau \right]$$
$$+ \int_{t_0}^{t} \exp\ \left[\int_{s}^{t} F(\tau)\ d\tau \right]\ G(s)\ Q_t(s)\ G^{\mathrm{T}}(s)\ \exp\ \left[\int_{s}^{t} F^{\mathrm{T}}(\tau)\ d\tau \right]\ ds, \tag{4.113}$$

one can differentiate both sides with respect to t to obtain

$$\begin{aligned}\frac{d}{dt}P_{xx}(t) &= \frac{d}{dt}\left\{\exp\left[\int_{t_0}^{t} F(s)\,ds\right] P_{xx}(t_0)\,\exp\left[\int_{t_0}^{t} F(s)\,ds\right]\right\} \\ &\quad + \frac{d}{dt}\left\{\int_{t_0}^{t}\exp\left[\int_{s}^{t} F(\tau)\,d\tau\right] G(s)\,Q_t(s)\,G^{\mathrm{T}}(s)\right. \\ &\quad \left.\times\exp\left[\int_{s}^{t} F^{\mathrm{T}}(\tau)\,d\tau\right] ds\right\} \qquad (4.114) \\ &= F(t)\exp\left[\int_{t_0}^{t} F(s)\,ds\right] P(t_0)\,\exp\left[\int_{t_0}^{t} F^{\mathrm{T}}(s)\,ds\right] \\ &\quad + \exp\left[\int_{t_0}^{t} F(s)\,ds\right] P(t_0)\,\exp\left[\int_{t_0}^{t} F^{\mathrm{T}}(s)\,ds\right] F^{\mathrm{T}}(t) \\ &\quad + F(t)\left\{\int_{t_0}^{t}\exp\left[\int_{s}^{t} F(\tau)\,d\tau\right] G(s)Q(s)G^{\mathrm{T}}(s)\right. \\ &\quad \left.\times\exp\left[\int_{s}^{t} F^{\mathrm{T}}(\tau)\,d\tau\right] ds\right\} \\ &\quad + \left\{\exp\left[\int_{s}^{t} F(\tau)\,d\tau\right] G(s)Q(s)G^{\mathrm{T}}(s)\exp\left[\int_{s}^{t} F^{\mathrm{T}}(\tau)\,d\tau\right] ds\right\} F^{\mathrm{T}}(t) \\ &\quad + G(t)Q(t)G^{\mathrm{T}}(t) \qquad (4.115) \\ &= F(t)P_{xx}(t) + P_{xx}(t)F^{\mathrm{T}}(t) + G(t)Q_t(t)G^{\mathrm{T}}\,(t). \qquad (4.116)\end{aligned}$$

4.6.2.3 Discrete-Time Model For the generic discrete-time process model

$$x_k = \Phi_{k-1}x_{k-1} + G_{k-1}w_{k-1} \qquad (4.117)$$

$$\mathop{\mathrm{E}}_{w}\langle w_i w_j^{\mathrm{T}}\rangle = \begin{cases} 0, & i \neq j \\ Q_i, & i = j, \end{cases} \qquad (4.118)$$

propagation of the mean μ_k is modeled by Equation 4.103, so that the central difference and its outer product

$$x_k - \mu_k = \Phi_{k-1}[x_{k-1} - \mu_{k-1}] + G_{k-1}w_{k-1} \qquad (4.119)$$

$$\begin{aligned}[x_k - \mu_k][x_k - \mu_k]^{\mathrm{T}} &= \Phi_{k-1}[x_{k-1} - \mu_{k-1}][x_{k-1} - \mu_{k-1}]^{\mathrm{T}}\Phi_{k-1}^{\mathrm{T}} \\ &\quad + G_{k-1}w_{k-1}w_{k-1}^{\mathrm{T}}G_{k-1}^{\mathrm{T}} \\ &\quad + \Phi_{k-1}[x_{k-1} - \mu_{k-1}]w_{k-1}^{\mathrm{T}}G_{k-1}^{\mathrm{T}} \\ &\quad + G_{k-1}w_{k-1}[x_{k-1} - \mu_{k-1}]^{\mathrm{T}}\Phi_{k-1}^{\mathrm{T}}, \qquad (4.120)\end{aligned}$$

and the expected values

$$P_k \overset{\text{def}}{=} \underset{x,w}{\mathrm{E}} \langle [x_k - \mu_k][x_k - \mu_k]^{\mathrm{T}} \rangle \tag{4.121}$$

$$\begin{aligned} &= \underset{x}{\mathrm{E}} \langle \Phi_{k-1}[x_{k-1} - \mu_{k-1}][x_{k-1} - \mu_{k-1}]^{\mathrm{T}} \Phi_{k-1}^{\mathrm{T}} \rangle \\ &\quad + \underset{w}{\mathrm{E}} \langle G_{k-1} w_{k-1} w_{k-1}^{\mathrm{T}} G_{k-1}^{\mathrm{T}} \rangle \\ &\quad + \underset{x}{\mathrm{E}} \langle \Phi_{k-1}[x_{k-1} - \mu_{k-1}] \rangle \underbrace{\underset{w}{\mathrm{E}} \langle w_{k-1}^{\mathrm{T}} G_{k-1}^{\mathrm{T}} \rangle}_{=0} \\ &\quad + \underbrace{\underset{w}{\mathrm{E}} \langle G_{k-1} w_{k-1} \rangle}_{=0} \underset{x}{\mathrm{E}} \langle [x_{k-1} - \mu_{k-1}]^{\mathrm{T}} \Phi_{k-1}^{\mathrm{T}} \rangle \end{aligned} \tag{4.122}$$

$$= \Phi_{k-1} P_{k-1} \Phi_{k-1}^{\mathrm{T}} + G_{k-1} Q_{k-1} G_{k-1}^{\mathrm{T}}. \tag{4.123}$$

4.6.3 Steady-State Solutions

One of the remarkable features of the Kalman filter is that the covariance of estimation uncertainty *with measurements* can have a finite steady-state value even if the covariance equation without sampling is unstable.

The issue considered here is whether the covariance equations *without measurement* are stable, in that their solutions approach a finite constant value as $t \to +\infty$ (continuous-time model) or $k \to +\infty$ (discrete-time model).

4.6.3.1 Continuous Time A steady-state solution for the covariance is a constant value which the covariance approaches as $t \to +\infty$, independent of the initial value of the covariance. Some time-varying and time-invariant systems may have steady-state solutions, but not all time-varying or time-invariant systems have steady-state solutions.

The time-invariant continuous-time model with zero-mean white noise

$$\dot{x}(t) = Fx(t) + w(t) \tag{4.124}$$

$$\underset{w}{\mathrm{E}} \langle w(t) w^{\mathrm{T}}(t) \rangle = Q_t \tag{4.125}$$

will have a steady-state covariance solution only if the steady-state equation

$$0 = \lim_{t \to +\infty} \frac{d}{dt} P(t) \tag{4.126}$$

$$= FP(\infty) + P(\infty) F^{\mathrm{T}} + Q_t \tag{4.127}$$

has a nonnegative definite solution $P(\infty)$. As a rule, it will have such solutions only if the eigenvalues of F have negative real parts.

4.6.3.2 Discrete Time The equivalent steady-state covariance equation in discrete time would be

$$P_\infty = \Phi P_\infty \Phi^{\mathrm{T}} + Q, \tag{4.128}$$

which may or may not have a solution, depending on the eigenvalues of Φ. As a rule, it will have a solution only if all the eigenvalues of Φ are inside the unit circle in the complex plane.

4.6.4 Results

The main results of this section are summarized in Table 4.3.

Example 4.7 (Scalar System in Continuous Time) A dynamic model is given by the scalar differential equation

$$\begin{aligned} \dot{x}(t) &= -x(t) + w(t) \\ \mathrm{E}x(0) &= 0 \\ P(0) &= 0 \\ F &= -1 \\ G &= 1 \\ w(t) &\equiv 1 \\ Q(t) &= 1. \end{aligned}$$

TABLE 4.3 Propagating Essential Moments with White-Noise Inputs

Continuous time equation	$\dot{x}(t) = F(t)x(t) + G(t)w(t)$
Mean value equation	$\frac{d}{dt}\mathrm{E}\langle x(t)\rangle = F(t)\mathrm{E}\langle x(t)\rangle$
Initial value	$\mathrm{E}\langle x(t_o)\rangle$
covariance equation	$\dot{P}(t) = F(t)P(t) + P(t)F^{\mathrm{T}}(t) + G(t)Q_t(t)G^{\mathrm{T}}(t)$
Initial value	$P(t_0)$
steady-state equation	$0 = FP(\infty) + P(\infty)F^{\mathrm{T}} + GQ_tG^{\mathrm{T}}$
(algebraic equation)	(for time-invariant model only)
Discrete-time equation	$x_k = \Phi_{k-1}x_{k-1} + G_{k-1}w_{k-1}$
Mean value equation	$\mathrm{E}\langle x_k\rangle = \Phi_{k-1}\mathrm{E}\langle x_{k-1}\rangle$
Covariance equation	$P_k = \Phi_{k-1}P_{k-1}\Phi_{k-1}^{\mathrm{T}} + G_{k-1}Q_{k-1}G_{k-1}^{\mathrm{T}}$
Initial value	P_0
steady-state equation	$P_\infty = \Phi P_\infty \Phi^{\mathrm{T}} + GQG^{\mathrm{T}}$
(algebraic equation)	(for time-invariant model only)

From Table 4.3,

$$\begin{aligned} \mathrm{E}\dot{x}(t) &= -Ex(t) + 1 \\ \mathrm{E}x(0) &= 0 \\ \mathrm{E}x(t) &= 1 - e^{-t},\ t \geq 0. \end{aligned}$$

The covariance equation is then given by

$$\begin{aligned} \dot{P}(t) &= -2P(t) + 1 \\ P(0) &= 0 \\ P(t) &= e^{-2t}P(0) + \int_0^t e^{-2(t-\tau)}1 \quad d\tau \\ &= \frac{1}{2}(1 - e^{-2t}). \end{aligned}$$

The steady-state covariance is then

$$\begin{aligned} P(\infty) &= \lim_{t\to+\infty} \frac{1}{2}(1 - e^{-2t}) \\ &= \frac{1}{2}. \end{aligned}$$

Example 4.8 (Scalar System in Discrete Time) A discrete-time model is given as

$$\left.\begin{array}{rcll} x_k^1 & = & -x_{k-1}^1 + w_{k-1}^1 & \\ \mathrm{E}w_{k-1}^1 & = & 0 & \text{(non-white)} \\ \Psi_w^1 & = & e^{-(k_1-k_2)} & \text{(autocorrelation)} \end{array}\right\} (a)$$

The model of non-white noise evolves into scalar difference equation (given in Table 4.1), called a shaping filter:

$$x_k^2 = \mathrm{e}^{-1}x_{k-1}^2 + \sqrt{1 - e^{-1}} \qquad w_{k-1} \quad (b),$$

where w_{k-1} is white noise with zero mean and covariance equal to 1.

Combining equation (a) and (b), as given by Equation 4.98,

$$x_k = \begin{bmatrix} -1 & 1 \\ 0 & e^{-1} \end{bmatrix} x_{k-1} + \begin{bmatrix} 0 \\ \sqrt{1 - e^{-1}} \end{bmatrix} w_{k-1}$$

$$P_k = \begin{bmatrix} -1 & 1 \\ 0 & e^{-1} \end{bmatrix} \quad P_{k-1} \begin{bmatrix} -1 & 0 \\ 1 & e^{-1} \end{bmatrix} + \begin{bmatrix} 0 \\ \sqrt{1 - e^{-1}} \end{bmatrix} 1 \begin{bmatrix} 0 & \sqrt{1 - e^{-1}} \end{bmatrix}$$

$$P_k^{11} = +P_{k-1}^{11} - 2P_{k-1}^{12} + P_k^{22}$$

$$P_k^{12} = -e^{-1} P_{k-1}^{12} + e^{-1} P_{k-1}^{22}$$
$$P_k^{22} = e^{-2} P_{k-1}^{22} + (1 - e^{-1})$$
$$P_0 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}.$$

The steady-state solution will be

$$P_\infty^{22} = \frac{(1 - e^{-1})}{1 - e^{-2}}$$
$$P_\infty^{12} = \frac{\left[e^{-1} \frac{(1-e^{-1})}{1-e^{-2}}\right]}{1 + e^{-1}}$$
$$P_\infty^{11} = \text{underdetermined.}$$

Example 4.9 (Steady-State Covariance of a Harmonic Resonator) The stochastic system model

$$\dot{x}(t) = Fx(t) + w(t),$$
$$\mathrm{E}\,\langle w(t_1) w^{\mathrm{T}}(t_2)\rangle = \delta(t_1 - t_2) Q,$$
$$Q = \begin{bmatrix} 0 & 0 \\ 0 & q \end{bmatrix}$$

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 2.7 has as its dynamic coefficient matrix the 2×2 constant matrix

$$F = \begin{bmatrix} 0 & 1 \\ -\frac{k_s}{m} & -\frac{k_d}{m} \end{bmatrix}$$
$$= \begin{bmatrix} 0 & 1 \\ -\omega_r^2 - \omega_d^2 & -2\omega_d \end{bmatrix},$$

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

$$\omega_r = \sqrt{\frac{k_s}{m} - \frac{k_d^2}{4m^2}} \text{ (undamped resonant frequency),}$$

$$\tau_d = \frac{2m}{k_d} \text{ (damping time constant)},$$

$$\omega_d = \frac{1}{\tau} \text{ (damping frequency)}.$$

If the covariance matrix $P(t)$ reaches some steady-state value $P(\infty)$ as $t \to +\infty$, the asymptotic covariance dynamic equation becomes

$$\begin{aligned} 0 &= \lim_{t\to\infty} \frac{d}{dt} P(t) \\ &= FP(\infty) + P(\infty)F^{\mathrm{T}} + Q, \end{aligned}$$

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 4.129 is equivalent to three scalar linear equations:

$$\begin{aligned} 0 &= \sum_{k=1}^{2} f_{1k} p_{k1} + \sum_{k=1}^{2} p_{1k} f_{k1} + q_{11}, \\ 0 &= \sum_{k=1}^{2} f_{1k} p_{k2} + \sum_{k=1}^{2} p_{1k} f_{k2} + q_{12}, \\ 0 &= \sum_{k=1}^{2} f_{2k} p_{k2} + \sum_{k=1}^{2} p_{2k} f_{k2} + q_{22}, \end{aligned}$$

with known parameters

$$\begin{array}{llllllll} q_{11} &= 0, & q_{12} &= 0, & q_{22} &= q, & & \\ f_{11} &= 0, & f_{12} &= 1, & f_{21} &= -\omega_r^2 - \omega_d^2, & f_{22} &= -2\omega_d. \end{array}$$

The linear system of equations above can be rearranged into the nonsingular 3×3 system of equations

$$\begin{bmatrix} 0 \\ 0 \\ q \end{bmatrix} = -\begin{bmatrix} 0 & 2 & 0 \\ -(\omega_r^2 + \tau_d^{-2}) & -\frac{2}{\tau_d} & 1 \\ 0 & -2(\omega_r^2 + \tau_d^{-2}) & -\frac{4}{\tau_d} \end{bmatrix} \begin{bmatrix} p_{11} \\ p_{12} \\ p_{22} \end{bmatrix}$$

with solution

$$\begin{bmatrix} p_{11} \\ p_{12} \\ p_{22} \end{bmatrix} = q \begin{bmatrix} \frac{\tau_d}{4(\omega_r^2 + \tau_d^{-2})} \\ 0 \\ \frac{\tau_d}{4} \end{bmatrix},$$

$$P(\infty) = \begin{bmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{bmatrix} = \frac{q\tau_d}{4(\omega_r^2 + \tau_d^{-2})} \begin{bmatrix} 1 & 0 \\ 0 & (\omega_r^2 + \tau_d^{-2}) \end{bmatrix}.$$

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. 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.

4.7 RELATIONSHIPS BETWEEN MODEL PARAMETERS

4.7.1 Parameters of Continuous and Discrete Models

The mathematical forms of the application models used in Kalman filtering (in discrete time) and Kalman–Bucy filtering (in continuous time) are summarized in Table 4.4. 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 (already shown in the table) is derived in Chapter 2, and the solution 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 4.131.

4.7.1.1 Sensor Models The last column of Table 4.4 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. 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 already 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 4.7.3, we will derive the $R(t) \rightarrow R_k$ relationship for some common types of anti-aliasing filters.

4.7.1.2 The Utility of Continuous-Time Models These relationships 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 the filter derivation can usually be reduced by following the maxim:

Think continuously. Act discretely.

TABLE 4.4 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	$F(t) = \frac{\partial \dot{x}}{\partial x}$	$H(t)\frac{\partial z}{\partial x}$
	$Q(t) = \mathrm{E}\,\langle w(t) w^{\mathrm{T}}(t)\rangle$	$R(t) = \mathrm{E}\,\langle v(t) v^{\mathrm{T}}(t)\rangle$
Kalman	$x_k = \Phi_{k-1}\, x_{k-1} + w_{k-1}$	$z_k = H_k\, x_k + v_k$
Parameters	$\Phi_{k-1} = \exp\left(\int_{t_{k-1}}^{t_k} F(s)\, ds\right)$	$H_k = H(t_k)$
	$Q_{k-1} = \mathrm{E}\,\langle w_{k-1} w_{k-1}^{\mathrm{T}}\rangle$	$R_k = \mathrm{E}\,\langle v_k v_k^{\mathrm{T}}\rangle$

4.7.2 Relationship between $Q(t)$ and Q_{k-1}

If we let $t = t_k$ and $t_0 = t_{k-1}$, then Equation 4.112 becomes

$$P_k = \Phi_{k-1} P_{k-1} \Phi_{k-1} + G_{k-1} Q_{k-1} G_{k-1}^{\mathrm{T}} \tag{4.129}$$

$$G_{k-1} Q_{k-1} G_{k-1}^{\mathrm{T}} = \int_{t_{k-1}}^{t_k} \exp\left[\int_s^{t_k} F(\tau)\, d\tau\right] G(s)\, Q_t(s)\, G^{\mathrm{T}}(s) \times \exp\left[\int_s^{t_k} F(\tau)\, d\tau\right]^{\mathrm{T}} ds, \tag{4.130}$$

where the last equation defines the relationship between the equivalent white-noise processes covariances in continuous time (Q_t) and discrete time (Q_k). In the case that $G_{k-1} = I = G(s)$, this becomes

$$Q_{k-1} = \int_{t_{k-1}}^{t_k} \exp\left[\int_s^{t_k} F(\tau)\, d\tau\right] Q_t(s)\, \exp\left[\int_s^{t_k} F(\tau)\, d\tau\right]^{\mathrm{T}} ds. \tag{4.131}$$

Solutions for some time-invariant system models are shown in Table 4.5

4.7.2.1 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 4.10 (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 the process noise in its *continuous-time model*

TABLE 4.5 Equivalent Constant Parameters, Continuous- and Discrete-Time models

Continuous Time		Discrete Time	
F	Q_t	Φ_k	Q_k
$[0]$	$[q]$	$[1]$	$[q\ \Delta t]$
$[-1/\tau]$	$[q]$	$[\exp\ (-\Delta t/\tau)]$	$\left[\frac{q\tau\ \left[1-\exp\ (-2\ \Delta t/\tau)\right]}{2}\right]$
$\begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix}$	$\begin{bmatrix}0 & 0\\ 0 & q\end{bmatrix}$	$\begin{bmatrix}1 & \Delta t\\ 0 & 1\end{bmatrix}$	$q\begin{bmatrix}\frac{(\Delta t)^3}{3} & \frac{(\Delta t)^2}{2}\\ \frac{(\Delta t)^2}{2} & \Delta t\end{bmatrix}$
$\begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 1\\ 0 & 0 & 0\end{bmatrix}$	$\begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & q\end{bmatrix}$	$\begin{bmatrix}1 & \Delta t & \frac{(\Delta t)^2}{2}\\ 0 & 1 & \Delta t\\ 0 & 0 & 1\end{bmatrix}$	$q\begin{bmatrix}\frac{(\Delta t)^5}{20} & \frac{(\Delta t)^4}{8} & \frac{(\Delta t)^3}{6}\\ \frac{(\Delta t)^4}{8} & \frac{(\Delta t)^3}{3} & \frac{(\Delta t)^2}{2}\\ \frac{(\Delta t)^3}{6} & \frac{(\Delta t)^2}{2} & \Delta t\end{bmatrix}$

$$\frac{d}{dt}x(t) = Fx(t) + w(t),$$
$$\mathrm{E}\,\langle w(t_1)\ w^{\mathrm{T}}(t_2)\rangle = \delta(t_1 - t_2)Q,$$
$$Q = \begin{bmatrix}0 & 0\\ 0 & q\end{bmatrix},$$

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 4.2 and 4.9.)

Following the derivation of Example 2.6, the fundamental solution matrix for the *unforced* dynamic system model can be expressed in the form

$$\begin{aligned}\Phi(t) &= e^{Ft}\\ &= e^{-t/\tau}\begin{bmatrix}\frac{S(t)+C(t)\omega\tau}{\omega\tau} & \frac{S(t)}{\omega}\\ -\frac{S(t)(1+\omega^2\tau^2)}{\omega\tau^2} & \frac{-S(t)+C(t)\omega\tau}{\omega\tau}\end{bmatrix},\\ S(t) &= \sin(\omega t),\\ C(t) &= \cos(\omega t),\end{aligned}$$

in terms of its resonant frequency ω and damping time constant τ. Its matrix inverse

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

at time $t = s$. Consequently, the indefinite integral matrix

$$\begin{aligned}
\Psi(t) &= \int_0^t \Phi^{-1}(s) \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \Phi^{\mathrm{T}-1}(s)\, ds \\
&= \frac{q}{\omega^2\tau^2} \int_0^t \begin{bmatrix} \tau^2 S(s)^2 & -\tau S(s)[\omega\tau C(s) + S(s)] \\ -\tau S(s)[\omega\tau C(s) + S(s)] & [\omega\tau C(s) + S(s)]^2 \end{bmatrix} e^{2s/\tau}\, ds \\
&= \begin{bmatrix} \dfrac{q\tau\{-\omega^2\tau^2 + [2S(t)^2 - 2C(t)\omega S(t)\tau + \omega^2\tau^2]\zeta^2\}}{4\omega^2(1+\omega^2\tau^2)} & \dfrac{-qS(t)^2\zeta^2}{2\omega^2} \\ \dfrac{-qS(t)^2\zeta^2}{2\omega^2} & \dfrac{q\{-\omega^2\tau^2 + [2S(t)^2 + 2C(t)\omega S(t)\tau + \omega^2\tau^2]\zeta^2\}}{4\omega^2\tau} \end{bmatrix}, \\
\zeta &= e^{t/\tau}.
\end{aligned}$$

The discrete-time covariance matrix Q_{k-1} can then be evaluated as (see Section 4.7)

$$\begin{aligned}
Q_{k-1} &= \Phi(\Delta t)\Psi(\Delta t)\Phi^{\mathrm{T}}(\Delta t) \\
&= \begin{bmatrix} q_{11} & q_{12} \\ q_{21} & q_{22} \end{bmatrix}, \\
q_{11} &= \frac{q\tau\{\omega^2\tau^2(1 - e^{-2\Delta t/\tau}) - 2S(\Delta t)e^{-2\Delta t/\tau}[S(\Delta t) + \omega\tau C(\Delta t)]\}}{4\omega^2(1+\omega^2\tau^2)}, \\
q_{12} &= \frac{qe^{-2\Delta t/\tau}S(\Delta t)^2}{2\omega^2}, \\
q_{21} &= q_{12}, \\
q_{22} &= \frac{q\{\omega^2\tau^2(1 - e^{-2\Delta t/\tau}) - 2S(\Delta t)e^{-2\Delta t/\tau}[S(\Delta t) - \omega\tau C(\Delta t)]\}}{4\omega^2\tau}.
\end{aligned}$$

Note that the *structure* of the discrete-time process noise covariance Q_{k-1} for this example is quite different from the continuous-time process noise Q. In particular, Q_{k-1} is a full matrix, although Q is a sparse matrix.

4.7.2.2 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:

$$Q_k = \sum_{i=1}^{\infty} \frac{Q^i \Delta t^i}{i!}. \tag{4.132}$$

Consider the Taylor series expansion of Q_k about t_{k-1}, where

$$Q^i = \left.\frac{d^i Q}{dt^i}\right|_{t=t_{k-1}},$$

$$\begin{aligned}
\dot{Q} &= FQ_k + Q_k F^{\mathrm{T}} + GQ(t)G^{\mathrm{T}}, \\
Q^{(1)} &= \dot{Q}(t_{k-1}) = GQ(t)G^{\mathrm{T}} \text{ since } Q(t_{k-1}) = 0, \\
Q^{(2)} &= \ddot{Q}(t_{k-1}) = F\dot{Q}(t_{k-1}) + \dot{Q}(t_{k-1})F^{\mathrm{T}}, \\
&= FQ^{(1)} + Q^{(1)}F^{\mathrm{T}}, \\
&\vdots \\
Q^{(i)} &= FQ^{(i-1)} + Q^{(i-1)}F^{\mathrm{T}}, \quad i = 1, 2, 3, \ldots
\end{aligned}$$

Taking only first-order terms in the above series,

$$Q_k \approx GQ(t)G^{\mathrm{T}}\Delta t. \tag{4.133}$$

This is not always a good approximation, as is shown in the following example.

Example 4.11 (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 MS 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 4.12 (Q_k Versus Q_t 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_t in continuous time can be derived in closed form. Starting with the continuous-time model

$$\dot{x} = \frac{-1}{\tau}x(t) + w(t), \text{ for } w(t) \in \mathcal{N}(0, Q_t),$$

$$F = \frac{-1}{\tau}$$

$$\Phi_k = \Phi(\Delta t) = \exp\left(\int_0^{\Delta t} F\, ds\right) = \exp\,(-\Delta t/\tau)$$

$$Q_k = \Phi_k^2 \int_0^{\Delta t} \Phi^{-1}(s) Q_t \Phi^{-\mathrm{T}}(s)\, ds = \frac{\tau}{2}[1 - \exp\,(-2\,\Delta t/\tau)]Q_t$$

$$Q_t = \frac{2}{\tau[1 - \exp\,(-2\,\Delta t/\tau)]} Q_k.$$

4.7.2.3 Van Loan's Method for Computing Q_k from Continuous Q For the general linear time-invariant model in continuous time

$$\dot{x}(t) = Fx(t) + Gw(t) \text{ with } w(t) \in \mathcal{N}(0, Q_t),$$

and with n state variables, form the $2n \times 2n$ matrix

$$M = \Delta t \begin{bmatrix} -F & GQ_tG^{\mathrm{T}} \\ 0 & F^{\mathrm{T}} \end{bmatrix}, \tag{4.134}$$

partitioned into a 2×2 array of $n \times n$ blocks.

A 1978 article by Van Loan [14] 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

$$\exp(M) = \begin{bmatrix} \Psi & \Phi_k^{-1}Q_k \\ 0 & \Phi^{\mathrm{T}} \end{bmatrix}, \tag{4.135}$$

where Ψ is not used, but 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 software 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` $\stackrel{\text{def}}{=} Q_t$.

Van Loan's method is coded in the MATLAB function `VanLoan` on the Wiley web site.

Example 4.13 (Using Van Loan's Method) Let the continuous-time model for a linear time-invariant stochastic system have parameter values

$$F = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, GQ_tG^{\mathrm{T}} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

and its equivalent implementation in discrete time require time-step $\Delta t = 1$. Then

$$M = \Delta t \begin{bmatrix} -F & GQ_tG^{\mathrm{T}} \\ 0 & F^{\mathrm{T}} \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$\begin{aligned} \exp(M) &= I + M + \frac{1}{2!}M^2 + \frac{1}{3!}M^3 + \frac{1}{4!}M^4 + \cdots \\ &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} + \frac{1}{2}\begin{bmatrix} 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \\ &\quad + \frac{1}{6}\begin{bmatrix} 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} + \frac{1}{24}\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} + \text{more zero terms} \\ &= \begin{bmatrix} 1 & -1 & -\frac{1}{6} & -\frac{1}{2} \\ 0 & 1 & \frac{1}{2} & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} \Psi & \Phi_k^{-1}Q_k \\ 0 & \Phi^{\mathrm{T}} \end{bmatrix}, \end{aligned}$$

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

$$\Phi_k = \exp\ (\Delta t\ F)$$

$$= I + \Delta t\, F + \frac{1}{2!}\underbrace{(\Delta t\, F)^2}_{=0} + \cdots$$

$$= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix},$$

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
```

4.7.3 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 white noise. If not, then the same noise data (or its PSD) can be used to devise a suitable shaping filter (i.e., an auxiliary noise model as a stochastic system with 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 by making it a system state variable, to be estimated by a Kalman filter. This approach also works for biases that are not constant.

4.7.3.1 Integrating Sensors Even if $R(t)$ and R_k are already zero-mean 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

$$v_k = \int_{t_{k-1}}^{t_k} v(t)\, dt, \tag{4.136}$$

$$R_k = \overline{R} \tag{4.137}$$

$$= \frac{1}{t_k - t_{k-1}} \int_{t_{k-1}}^{t_k} R(t)\, dt, \tag{4.138}$$

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).

4.7.3.2 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 amount of 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.

4.8 ORTHOGONALITY PRINCIPLE

4.8.1 Estimators Minimizing Expected Quadratic Loss Functions

A block diagram representation of an estimator of the state of a system represented by Equation 4.64 is shown in Figure 4.6. 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 RV $X(t)$ and an "estimate" $\hat{x}(t)$.

A quadratic "loss" function of the estimation error has the form

$$[x(t) - \hat{x}(t)]^{\mathrm{T}} M [x(t) - \hat{x}(t)], \tag{4.139}$$

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 $X(t)$

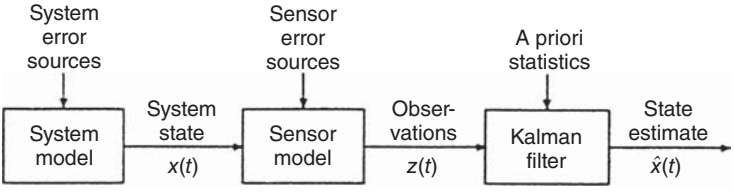


Figure 4.6 Block diagram of estimator model.

(minimizing the average of a quadratic cost function) is the conditional expectation of $X(t)$ given the observation $z(t)$:

$$\hat{x} = \mathrm{E}\,\langle x(t)|z(t)\rangle \quad \text{minimizes}$$
$$\mathrm{E}\,\langle [x(t)-\hat{x}(t)]^{\mathrm{T}} M[x(t)-\hat{x}(t)]|z(t)\rangle. \tag{4.140}$$

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 4.140 assumes the form

$$\hat{x}(t_2) = \mathrm{E}\,\langle x(t_2)|z(t), 0 \le t \le t_1\rangle \tag{4.141}$$

and the corresponding equation for a similar discrete model is

$$\hat{x}_{k_2} = \mathrm{E}\,\langle \hat{x}_{k_2}|z_1, z_2, \ldots, z_{k_1}\rangle, \quad 1 \le k_2 \le k_1. \tag{4.142}$$

Let

$$J = \mathrm{E}\,\langle [x(t)-\hat{x}(t)]^{\mathrm{T}} M[x(t)-\hat{x}(t)]|z(t)\rangle. \tag{4.143}$$

Recall that $\hat{x}(t)$ is a nonrandom function of the observations

$$0 = \frac{dJ}{d\hat{x}} \tag{4.144}$$
$$= -2M\,\mathrm{E}\,\langle [x(t)-\hat{x}(t)]|z(t)\rangle, \tag{4.145}$$
$$\mathrm{E}\,\langle \hat{x}(t)|z(t)\rangle = \hat{x}(t) = \mathrm{E}\,\langle x(t)|z(t)\rangle. \tag{4.146}$$

This proves the result of Equation 4.140. If $X(t)$ and $z(t)$ are jointly normal (Gaussian), the nonlinear minimum variance and linear minimum variance estimators coincide

$$\mathrm{E}\,\langle x_{k_2}|z_1, z_2, \ldots, z_{k_1}\rangle = \sum_{i=1}^{k_1} \alpha_i z_i \tag{4.147}$$

and

$$\mathrm{E}\,\langle x(t_2)|z(t), 0 \le t \le t_1\rangle = \int_0^{t_1} \alpha(t,\tau) z(\tau)\, d\tau. \tag{4.148}$$

4.8.1.1 Proof for the Discrete Case Let the probability density

$$p[x_{k_2}|z_{k_1}] \tag{4.149}$$

be Gaussian and let $\alpha_1, \alpha_2, \ldots, \alpha_{k_1}$ satisfy

$$\mathrm{E}\left\langle \left[x_{k_2} - \sum_{i=1}^{k_1} \alpha_i z_i \right] z_j^{\mathrm{T}} \right\rangle = 0, \quad j = 1, \ldots, k_1, \tag{4.150}$$

and

$$k_1 < k_2, \quad k_1 = k_2, \quad \text{or} \quad k_1 > k_2. \tag{4.151}$$

The existence of vectors α_i satisfying this equation is guaranteed because the covariance $[z_i, z_j]$ is nonsingular.

The vectors

$$\left[x_{k_2} - \sum \alpha_i z_i\right] \tag{4.152}$$

and z_i are independent. Then it follows from the zero-mean property of the sequence x_k that

$$\begin{aligned} \mathrm{E}\left\langle \left[x_{k_2} - \sum_{i=1}^{k_1} \alpha_i z_i\right] \middle| z_1, \ldots, z_{k_1} \right\rangle &= \mathrm{E}\left\langle x_{k_2} - \sum_{i+1}^{k_1} \alpha_i z_i \right\rangle \\ &= 0, \end{aligned}$$

$$\mathrm{E}\left\langle x_{k_2} | z_1, z_2, \ldots, z_{k_1} \right\rangle = \sum_{i=1}^{k_1} \alpha_i z_i.$$

The proof of the continuous case is similar.

The linear minimum variance estimator is unbiased, that is,

$$\mathrm{E}\left\langle x(t) - \hat{x}(t) \right\rangle = 0, \tag{4.153}$$

where

$$\hat{x}(t) = \mathrm{E}\left\langle x(t) | z(t) \right\rangle. \tag{4.154}$$

In other words, an unbiased estimate is one whose expected value is the same as that of the quantity being estimated.

4.8.2 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 M be a 1×1 weighting matrix. The constants α_0 and α_1 that minimize the *mean-squared* (MS) *error*

$$e = \mathrm{E}\left\langle [x - (\alpha_0 + \alpha_1 z)]^2 \right\rangle = \int_{\infty}^{\infty} \int_{\infty}^{\infty} [x - (\alpha_0 + \alpha_1 z)]^2 p(x, z)\, dx\, dz \tag{4.155}$$

are given by

$$\begin{aligned} \alpha_1 &= \frac{r\sigma_x}{\sigma_z}, \\ \alpha_0 &= \mathrm{E}\left\langle x \right\rangle - \alpha_1 \mathrm{E}\left\langle z \right\rangle, \end{aligned}$$

and the resulting minimum MS error $e_{\min}$ is

$$e_{\min} = \sigma_x^2 (1 - r^2), \tag{4.156}$$

where the ratio

$$r = \frac{\mathrm{E}\,\langle xz\rangle}{\sigma_x \sigma_z} \tag{4.157}$$

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

$$\frac{d}{d\alpha_0}\mathrm{E}\,\langle [x - \alpha_0 - \alpha_1 z]^2\rangle = 0 \tag{4.158}$$

and

$$\alpha_0 = \mathrm{E}\,\langle x\rangle - \alpha_1 \mathrm{E}\,\langle z\rangle. \tag{4.159}$$

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

$$\begin{aligned}
0 &= \frac{d}{d\alpha_1}\mathrm{E}\,\langle [x - \alpha_0 - \alpha_1 z]^2\rangle \\
&= 2\alpha_1 \mathrm{E}\,\langle (z - \mathrm{E}\,\langle z\rangle)^2\rangle - 2\mathrm{E}\,\langle (x - \mathrm{E}\,\langle x\rangle)(z - \mathrm{E}\,\langle z\rangle)\rangle, \qquad (4.160) \\
\alpha_1 &= \frac{\mathrm{E}\,\langle (x - \mathrm{E}\,\langle x\rangle)(z - \mathrm{E}\,\langle z\rangle)\rangle}{\mathrm{E}\,\langle (z - \mathrm{E}\,\langle z\rangle)^2\rangle} \\
&= \frac{r\sigma_x \sigma_z}{\sigma_z^2} \\
&= \frac{r\sigma_x}{\sigma_z}, \qquad (4.161) \\
e_{\min} &= \sigma_x^2 - 2r^2\sigma_x^2 + r^2\sigma_x^2 \\
&= \sigma_x^2 (1 - r^2).
\end{aligned}$$

Note that if one assumes that x and z have zero means,

$$\mathrm{E}\,\langle x\rangle = \mathrm{E}\,\langle z\rangle = 0, \tag{4.162}$$

then we have the solution

$$\alpha_0 = 0. \tag{4.163}$$

Orthogonality Principle The constant α_1 that minimizes the MS error

$$e = \mathrm{E}\,\langle [x - \alpha_1 z]^2 \rangle \tag{4.164}$$

is such that $x - \alpha_1 z$ is *orthogonal* to z. That is,

$$\mathrm{E}\,\langle [x - \alpha_1 z] z \rangle = 0, \tag{4.165}$$

and the value of the minimum MS error is given by the formula

$$e_m = \mathrm{E}\,\langle (x - \alpha_1 z) x \rangle. \tag{4.166}$$

4.8.3 A Geometric Interpretation of Orthogonality

Consider all RVs 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

$$\mathrm{E}\,\langle x^2 \rangle = \mathrm{E}\,\langle x^{\mathrm{T}} x \rangle \tag{4.167}$$

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 4.7.

The MS 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,

$$\mathrm{E}\,\langle (x - \alpha_1 z) z \rangle = 0. \tag{4.168}$$

We will apply the orthogonality principle to derive Kalman estimators in Chapter 5.

4.9 SUMMARY

4.9.1 Important Points to Remember

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

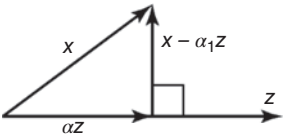


Figure 4.7 Orthogonality diagram.

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.

Random Variables 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 RP if the expected value of their pointwise product is zero.

4.9.2 Important Equations to Remember

The density function of an n-vector-valued (or *multivariate*) *Gaussian probability distribution* $\mathcal{N}(\overline{x}, P)$ has the functional form

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

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 RPs with certain types of spectral properties or temporal correlations.

PROBLEMS

4.1 For the discrete-time model

$$x_k = m^2 x_{k-1} + w_{k-1},\ Ew_{k-1} = 0,\ Q_{k-1} = \Delta(k-k),\ P_0 = 1, \text{ and } \mathop{\mathrm{E}}_{x}\langle x_0\rangle = 1,$$

find $E\langle x_3\rangle$, P_k, and P_∞.

4.2 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.

4.3 Let $R(T)$ be an arbitrary autocorrelation function for an 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.

4.4 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

ω = frequency in radians per second,

σ = root-mean-square (RMS) turbulence intensity,

L = scale length in feet, and

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,

$$L = 600 \text{ ft}$$

$$\sigma_v = 0.15 \text{ mean cross-wind (knots).}$$

4.5 Consider the RP

$$x(t) = \cos(\omega_0 t + \theta_1)\cos(\omega_0 t + \theta_2),$$

where θ_1 and θ_2 are independent RVs uniformly distributed between 0 and 2π.

(a) Show that $X(t)$ is WSS.

(b) Calculate $\psi_x(\tau)$ and $\Psi_x(\omega)$.

(c) Discuss the ergodicity of $X(t)$.

4.6 Let $\psi_x(\tau)$ be the autocorrelation of a WSS RP. Is the real part of $\psi_x(\tau)$ necessarily also an autocorrelation? If your answer is affirmative, prove it; if negative, give a counterexample.

4.7 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 $\psi_{xy}(\tau)$.

4.8 The RP $X(t)$ has mean zero and autocorrelation function

$$\psi_x(\tau) = e^{-|\tau|}.$$

Find the autocorrelation function for

$$y(t) = \int_0^t x(u)\, du, \quad t > 0.$$

4.9 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 spectral density 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 autocorrelation function?

$$\psi(\tau) = \begin{cases} 1 - |\tau|, & |\tau| < 1, \\ 0 & \text{otherwise.} \end{cases}$$

Explain.

4.10 Assume $X(t)$ is a stationary RP with autocorrelation function

$$\psi_x(\tau) = \begin{cases} 1 - |\tau|, & -1 \le \tau \le 1, \\ 0 & \text{otherwise.} \end{cases}$$

Find the spectral density $\Psi_y(\omega)$ 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 $[0, 2\pi]$.

4.11 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 autocorrelation function $\psi_y(t, s)$ for

$$y(t) = \int_0^t x(u)\, du.$$

4.12 Let ψ_1 and ψ_2 be two arbitrary continuous, absolutely integrable autocorrelation functions. Are the following necessarily autocorrelation 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 $\psi_1$with ψ_2).

4.13 Give a short reason for each answer:

(a) If $f(t)$ and $g(t)$ are autocorrelation functions, $f^2(t) + g(t)$ is (necessarily, perhaps, never) an autocorrelation function.

(b) As in (a), $f^2(t) - g(t)$ is (necessarily, perhaps, never) an autocorrelation 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 autocorrelation function.

(e) Let $X(t)$ be strictly stationary and ergodic and α be a Gaussian RV with mean 0 and variance 1 and α is independent of $X(t)$. Then $y(t) = \alpha x(t)$ is (necessarily, perhaps, never) ergodic.

4.14 Which of the following functions is an autocorrelation function of a WSS process? Give a brief reason for each answer.

(a) $e^{-|\tau|}$.

(b) $e^{-|\tau|}\cos\tau$.

(c) $\Gamma(t) = \begin{cases} 1, & |t| < a, \\ 0 & |t| \geq a. \end{cases}$

(d) $e^{-|\tau|} \sin \tau$.

(e) $\frac{3}{2}e^{-|\tau|} - e^{-2|\tau|}$.

(f) $2e^{-2|\tau|} - e^{-|\tau|}$.

4.15 Is the following function an autocorrelation function of a WSS process?

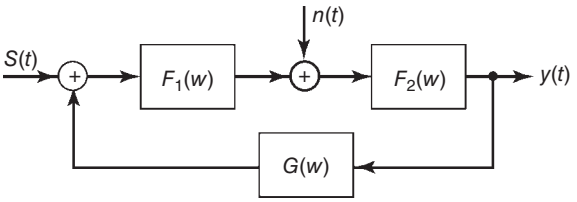
$$\psi_x(\tau) = 1.5e^{-|\tau|} + (11/3)e^{-3|\tau|}.$$

4.16 Discuss each of the following:

(a) The distinction between stationarity and wide-sense stationarity.

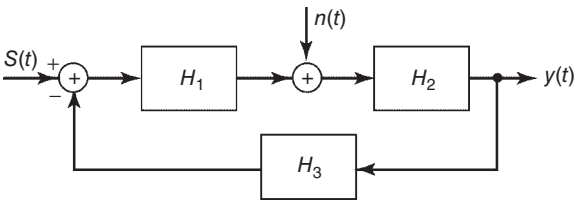
(b) The periodic character of the cross-correlation function of two processes that are themselves periodic with periods mT and nT, respectively.

4.17 A system transfer function can sometimes be experimentally determined by injecting white noise $w(t)$ and measuring the cross-correlation between the system output and the white noise. Here we consider the following system:



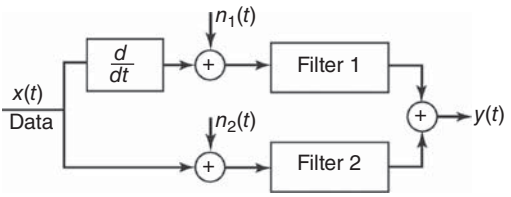
We assume $\Psi_s(\omega)$ known, $S(t)$ and $w(t)$ independent, and $\Psi_n(\omega) = 1$. Find $\Psi_{yn}(\omega)$. *Hint:* Write $y(t) = y_S(t) + y_n(t)$, where y_S and y_n are the parts of the output due to S and n, respectively.

4.18 Let $S(t)$ and $w(t)$ 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 $w(t)$ is zero and $n_0(t)$ is the output when $S(t)$ is zero. Find the output signal-to-noise ratio, defined as $\mathrm{E}\,\langle S_0^2(t)\rangle / \mathrm{E}\,\langle n_0^2(t)\rangle$.

4.19 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 pictured below:



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 MS 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 spectral densities Ψ_x, Ψ_{n_1}, and Ψ_{n_2}.

4.20 Let $X(t)$ be the solution of

$$\dot{x} + x = w(t),$$

where $w(t)$ is white noise with spectral density 2π.

(a) Assuming that the above system has been operating since $t = -\infty$, find $\psi_x(t_1 t_2)$. Investigate whether $X(t)$ is WSS, and if so, express ψ_x accordingly.

(b) Instead of the system in (a), consider

$$\dot{x} + x = \begin{cases} w(t), & t \geq 0, \\ 0, & t < 0, \end{cases}$$

where $x(0) = 0$. Again, compute $\psi_x(t_1, t_2)$.

(c) Let $y(t) = \int_0^t x(\tau) d\tau$. Find $\psi_{xy}(t_1, t_2)$ 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 which 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)*.

4.21 Let *x(t)* be the solution of

$$\dot{x} + x = w(t)$$

with initial condition $x(0) = x_0$. It is assumed that $w(t)$ is white noise with spectral density 2π and is turned on at $t = 0$. The initial condition x_0 is an RV independent of $w(t)$ and with 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.

4.22 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).

4.23 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.

4.24 For the continuous-time system

$$\dot{x}(t) = -x(t) + w(t)$$
$$x(0) = 1$$
$$P(0) = 1$$
$$w(t) = 2$$
$$Q(t) = e^{-2}\delta(t - \tau).$$

Find $\mathrm{E}\langle x(t)\rangle$, $P_x(t)$, P_∞.

4.25 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 $w_k \in N(0, 1)$ and white.

4.26 Find the steady-state covariance for the state-space model given in Example 3.4.

4.27 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 $F = Q = G = 1$.

4.28 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 $\Phi = Q = 1$.

4.29 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.

4.30 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 $\mathrm{E}(w(t_1) \quad w(t_2)) = e^{-|t_1 - t_2|}$. Assume the initial value of the covariance (P_0) is 1.

4.31 Suppose that *x(t)* has autocorrelation 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 MS error corresponding to the above. *Hint:* Use the orthogonality principle.

REFERENCES

[1] E. Allen, *Modeling with Itô Stochastic Differential Equations*, Springer, Berlin, 2007.

[2] L. Arnold, *Stochastic Differential Equations: Theory and Applications*, Kreiger, Malabar, FL, 1992.

[3] J. Baras and V. Mirelli, *Recent Advances in Stochastic Calculus*, Springer-Verlag, New York, 1990.

[4] K. Itô and H. P. McKean Jr., *Diffusion Processes and Their Sample Paths*, Academic Press, New York, 1965.

[5] B. Oksendal, *Stochastic Differential Equations: An Introduction with Applications*, Springer, Berlin, 2007.

[6] K. Sobczyk, *Stochastic Differential Equations with Applications to Physics and Engineering*, Kluwer Academic Press, Dordrecht, The Netherlands, 1991.

[7] R. L. Stratonovich, *Topics in the Theory of Random Noise* (R. A. Silverman, Ed.), Gordon & Breach, New York, 1963.

[8] 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.

[9] P. Langevin, "Sur la théory du muovement brownien," *Comptes Rendus de l'Academie des Sciences*, Vol. 146, pp. 530–533, Paris, 1908.

[10] R. S. Bucy and P. D. Joseph, *Filtering for Stochastic Processes with Applications to Guidance*, American Mathematical Society, Chelsea Publishing, Providence, RI, 2005.

[11] K. Itô, *Lectures on Stochastic Processes*, Tata Institute of Fundamental Research, Bombay (Mumbai), 1961.

[12] 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.

[13] 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.

[14] C. F. Van Loan, "Computing integrals involving the matrix exponential," *IEEE Transactions on Automatic Control*, Vol. AC-23, No. 3, pp. 395–404, 1978.

5

LINEAR OPTIMAL FILTERS AND PREDICTORS

Prediction is difficult—especially of the future.
—attributed to Niels Henrik David Bohr (1885–1962)

5.1 CHAPTER FOCUS

5.1.1 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.

We suppose that stochastic systems can be represented by the types of plant and measurement models (for continuous and discrete time) shown as Equations 5.1–5.6 in Table 5.1, with dimensions of the vector and matrix quantities as shown in Table 5.2. 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 random processes, and the initial value x_0 is a 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 5.5 will remove this restriction and modify the estimator equations accordingly.

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

Kalman Filtering: Theory and Practice Using MATLAB®, Fourth Edition.
Mohinder S. Grewal and Angus P. Andrews.
© 2015 John Wiley & Sons, Inc. Published 2015 by John Wiley & Sons, Inc.

TABLE 5.1 Linear Plant and Measurement Models

Model	Continuous Time	Discrete Time	Equation Number
Plant	$\dot{x}(t) = F(t)x(t) + w(t)$	$x_k = \Phi_{k-1}x_{k-1} + w_{k-1}$	5.1
Measurement	$z(t) = H(t)x(t) + v(t)$	$z_k = H_k x_k + v_k$	5.2
Plant noise	$\mathrm{E}\langle w(t)\rangle = 0$	$\mathrm{E}\langle w_k\rangle = 0$	5.3
	$\mathrm{E}\langle w(t)w^{\mathrm{T}}(s)\rangle = \delta(t-s)Q(t)$	$\mathrm{E}\langle w_k w_i^{\mathrm{T}}\rangle = \Delta(k-i)Q_k$	5.5
Observation noise	$\mathrm{E}\langle v(t)\rangle = 0$	$\mathrm{E}\langle v_k\rangle = 0$	5.5
	$\mathrm{E}\langle v(t)v^{\mathrm{T}}(s)\rangle = \delta(t-s)R(t)$	$\mathrm{E}\langle v_k v_i^{\mathrm{T}}\rangle = \Delta(k-i)R_k$	5.6

TABLE 5.2 Dimensions of Arrays in Linear Model

Symbol	Dimensions	Symbol	Dimensions
x, w	$n \times 1$	Φ, Q	$n \times n$
z, v	$\ell \times 1$	H	$\ell \times n$
R	$\ell \times \ell$	Δ, δ	Scalar

$$\mathrm{E}\langle [x_k - \hat{x}_k]^{\mathrm{T}} M [x_k - \hat{x}_k]\rangle,$$

where M is any *symmetric nonnegative-definite weighting matrix*.

5.1.2 Main Points to Be Covered

5.1.2.1 Linear Least-Mean-Squared 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* (LQ) estimation problem. The dynamic systems are linear and the performance cost functions are quadratic (least-mean-squared estimation error). We have already seen in Chapter 3 that the solution does not depend on the random processes being Gaussian.

5.1.2.2 Filtering, Prediction, and Smoothing There are three general types of estimators for the LQ 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 *beyond* the time that the state of the dynamic system is to be estimated:

$$t_{\text{obs}} > t_{\text{est}}.$$

5.1.2.3 Kalman gain A straightforward and simple approach using the orthogonality principle is used in the derivation[1] of *estimators*. These estimators will have *minimum variance* and be *unbiased* and *consistent*.

Two easier derivations of the Kalman gain formula are also provided, one starting with the Gaussian maximum-likelihood (ML) estimator and the other starting with the linear least-mean-square (LMS) estimator.

5.1.2.4 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 random 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.

5.1.2.5 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.

[1]For more mathematically oriented derivations, consult any of the references such as Anderson and Moore [1], Bozic [2], Brammer and Siffling [3], Brown [4], Bryson and Ho [5], Bucy and Joseph [6], Catlin [7], Chui and Chen [8], Gelb et al. [9], Jazwinski [10], Kailath [11], Maybeck [12, 13], Mendel [14, 15], Nahi [16], Ruymgaart and Soong [17], and Sorenson [18].

[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 [19], although not the form that we have here [20, 21]. 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 to initial-value problems [22].

5.2 KALMAN FILTER

5.2.1 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 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*.

5.2.2 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 5.8.1). 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:

$$\hat{x}_{k(+)} = K_k^1 \hat{x}_{k(-)} + \overline{K}_k z_k, \tag{5.7}$$

where $\hat{x}_{k(-)}$ is the a priori estimate of x_k and $\hat{x}_{k(+)}$ is the a posteriori value of the estimate.

5.2.3 Solving for the Kalman Gain

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 4.8.2. This orthogonality condition can be written in the form

$$\mathrm{E}\langle [x_k - \hat{x}_{k(+)}] z_i^{\mathrm{T}} \rangle = 0, \quad i = 1, 2, \ldots, k-1, \tag{5.8}$$

$$\mathrm{E}\langle [x_k - \hat{x}_{k(+)}] z_k^{\mathrm{T}} \rangle = 0. \tag{5.9}$$

If one substitutes the formula for x_k from Equation 5.1 (in Table 5.1) and for $\hat{x}_{k(+)}$ from Equation 5.7 into Equation 5.8, then one will observe from Equations 5.1 and 5.2 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 $\mathrm{E}\langle w_k z_i^{\mathrm{T}} \rangle = 0$ for $1 \le i \le k$. (See Problem 5.6.)

Using this result, one can obtain the following relation:

$$\mathrm{E}\langle [\Phi_{k-1} x_{k-1} + w_{k-1} - K_k^1 \hat{x}_{k(-)} - \overline{K}_k z_k] z_i^{\mathrm{T}} \rangle = 0, \quad i = 1, \ldots, k-1. \tag{5.10}$$

But because $z_k = H_k x_k + v_k$, Equation 5.10 can be rewritten as

$$\mathrm{E}\langle [\Phi_{k-1} x_{k-1} - K_k^1 \hat{x}_{k(-)} - \overline{K}_k H_k x_k - \overline{K}_k v_k] z_i^{\mathrm{T}} \rangle = 0, \quad i = 1, \ldots, k-1. \tag{5.11}$$

We also know that Equations 5.8 and 5.9 hold at the previous step, that is,

$$\mathrm{E}\langle[x_{k-1} - \hat{x}_{(k-1)(+)}]z_i^{\mathrm{T}}\rangle = 0, \quad i = 1, \ldots, k-1,$$

and

$$\mathrm{E}\langle v_k z_i^{\mathrm{T}}\rangle, = 0, \quad i = 1, \ldots, k-1.$$

Then Equation 5.11 can be reduced to the form

$$\begin{aligned}
\Phi_{k-1}\mathrm{E}\langle x_{k-1}z_i^{\mathrm{T}}\rangle - K_k^1\mathrm{E}\langle\hat{x}_{k(-)}z_i^{\mathrm{T}}\rangle - \overline{K}_k H_k\Phi_{k-1}\mathrm{E}\langle x_{k-1}z_i^{\mathrm{T}}\rangle - \overline{K}_k\mathrm{E}\langle v_k z_i^{\mathrm{T}}\rangle &= 0,\\
\Phi_{k-1}\mathrm{E}\langle x_{k-1}z_i^{\mathrm{T}}\rangle - K_k^1\mathrm{E}\langle\hat{x}_{k(-)}z_i^{\mathrm{T}}\rangle - \overline{K}_k H_k\Phi_{k-1}\mathrm{E}\langle x_{k-1}z_i^{\mathrm{T}}\rangle &= 0,\\
\mathrm{E}\langle[x_k - \overline{K}_k H_k x_k - K_k^1 x_k] - K_k^1(\hat{x}_{k(-)} - x_k)z_i^{\mathrm{T}}\rangle &= 0,\\
[I - K_k^1 - \overline{K}_k H_k]\mathrm{E}\langle x_k z_i^{\mathrm{T}}\rangle &= 0. \qquad (5.12)
\end{aligned}$$

Equation 5.12 can be satisfied for any given x_k if

$$K_k^1 = I - \overline{K}_k H_k. \qquad (5.13)$$

Clearly, this choice of K_k^1 causes Equation 5.7 to satisfy a portion of the condition given by Equation 5.8, which was derived in Section 5.8. The choice of $\overline{K}_k$ is such that Equation 5.9 is satisfied.

Let the errors

$$\tilde{x}_{k(+)} \stackrel{\Delta}{=} \hat{x}_{k(+)} - x_k, \qquad (5.14)$$

$$\tilde{x}_{k(-)} \stackrel{\Delta}{=} \hat{x}_{k(-)} - x_k, \qquad (5.15)$$

$$\begin{aligned}
\tilde{z}_k &\stackrel{\Delta}{=} \hat{z}_{k(-)} - z_k\\
&= H_k\hat{x}_{k(-)} - z_k. \qquad (5.16)
\end{aligned}$$

Vectors $\tilde{x}_{k(+)}$ and $\tilde{x}_{k(-)}$ are the estimation errors after and before updates, respectively.[3]

The parameter $\hat{x}_k$ depends linearly on x_k, which depends linearly on z_k. Therefore, from Equation 5.9,

$$\mathrm{E}\langle[x_k - \hat{x}_{k(+)}]\tilde{z}_{k(-)}^{\mathrm{T}}\rangle = 0 \qquad (5.17)$$

and also (by subtracting Equation 5.9 from Equation 5.17)

$$\mathrm{E}\langle[x_k - \hat{x}_{k(+)}]\tilde{z}_k^{\mathrm{T}}\rangle = 0. \qquad (5.18)$$

[3]The symbol ~ is officially called a *tilde* but often called a *squiggle*.

Substitute for $x_k, \hat{x}_{k(+)}$ and $\tilde{z}_k$ from Equations 5.1, 5.7, and 5.16, respectively. Then

$$\mathrm{E}\langle[\Phi_{k-1}x_{k-1} + w_{k-1} - K^1_{k(-)} - \overline{K}_k z_k][H_k\hat{x}_{k(-)} - z_k]^{\mathrm{T}}\rangle = 0.$$

However, by the system structure

$$\mathrm{E}\langle w_k z_k^{\mathrm{T}}\rangle = \mathrm{E}\langle w_k \hat{x}^{\mathrm{T}}_{k(+)}\rangle = 0,$$
$$\mathrm{E}\langle[\Phi_{k-1}x_{k-1} - K^1_k\hat{x}_{k(-)} - \overline{K}_k z_k][H_k\hat{x}_{k(-)} - z_k]^{\mathrm{T}}\rangle = 0.$$

Substituting for K^1_k, z_k, and $\tilde{x}_{k(-)}$ and using the fact that $\mathrm{E}\tilde{x}_{k(-)}v_k^{\mathrm{T}} = 0$, this last result can be modified as follows:

$$\begin{aligned}
0 &= \mathrm{E}\langle[\Phi_{k-1}x_{k-1} - \hat{x}_{k(-)} + \overline{K}_k H_k\hat{x}_{k(-)} - \overline{K}_k H_k x_k - \overline{K}_k v_k] \\
&\qquad [H_k\hat{x}_{k(-)} - H_k x_k - v_k]^{\mathrm{T}}\rangle \\
&= \mathrm{E}\langle[(x_k - \hat{x}_{k(-)}) - \overline{K}_k H_k(x_k - \hat{x}_{k(-)}) - \overline{K}_k v_k][H_k\tilde{x}_{k(-)} - v_k]^{\mathrm{T}}\rangle \\
&= \mathrm{E}\langle[(-\tilde{x}_{k(-)} + \overline{K}_k H_k\tilde{x}_{k(-)} - \overline{K}_k v_k][H_k\tilde{x}_{k(-)} - v_k]^{\mathrm{T}}\rangle.
\end{aligned}$$

By definition, the a priori covariance (the error covariance matrix before the update) is

$$P_{k(-)} = \mathrm{E}\langle\tilde{x}_{k(-)}\tilde{x}^{\mathrm{T}}_{k(-)}\rangle.$$

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 Kalman gain can be expressed as

$$\overline{K}_k = P_{k(-)}H_k^{\mathrm{T}}[H_k P_{k(-)}H_k^{\mathrm{T}} + R_k]^{-1}, \tag{5.19}$$

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

$$P_{k(+)} = \mathrm{E}\langle[\tilde{x}_{k(+)}\tilde{x}^{\mathrm{T}}_{k(+)}]\rangle. \tag{5.20}$$

By substituting Equation 5.13 into Equation 5.7, one obtains the equations

$$\begin{aligned}
\hat{x}_{k(+)} &= (I - \overline{K}_k H_k)\hat{x}_{k(-)} + \overline{K}_k z_k, \\
\hat{x}_{k(+)} &= \hat{x}_{k(-)} + \overline{K}_k[z_k - H_k\hat{x}_{k(-)}].
\end{aligned} \tag{5.21}$$

Subtract x_k from both sides of the latter equation to obtain the equations

$$\begin{aligned}\hat{x}_{k(+)} - x_k &= \hat{x}_{k(-)} + \overline{K}_k H_k x_k + \overline{K}_k v_k - \overline{K}_k H_k \hat{x}_{k(-)} - x_k,\\ \tilde{x}_{k(+)} &= \tilde{x}_{k(-)} - \overline{K}_k H_k \tilde{x}_{k(-)} + \overline{K}_k v_k,\\ \tilde{x}_{k(+)} &= (I - \overline{K}_k H_k)\tilde{x}_{k(-)} + \overline{K}_k v_k. \end{aligned} \tag{5.22}$$

By substituting Equation 5.22 into Equation 5.20 and noting that $\mathrm{E}\langle \tilde{x}_{k(-)} v_k^{\mathrm{T}} \rangle = 0$, one obtains

$$\begin{aligned}P_{k(+)} &= \mathrm{E}\langle [I - \overline{K}_k H_k]\tilde{x}_{k(-)}\tilde{x}_{k(-)}^{\mathrm{T}}[I - \overline{K}_k H_k]^{\mathrm{T}} + \overline{K}_k v_k v_k^{\mathrm{T}} \overline{K}_k^{\mathrm{T}} \rangle\\ &= (I - \overline{K}_k H_k) P_{k(-)} (I - \overline{K}_k H_k)^{\mathrm{T}} + \overline{K}_k R_k \overline{K}_k^{\mathrm{T}}. \end{aligned} \tag{5.23}$$

This last equation is the so-called "Joseph form" of the covariance update equation derived by Bucy and Joseph [6]. By substituting for $\overline{K}_k$ from Equation 5.19, it can be put into the following forms:

$$\begin{aligned}P_{k(+)} &= P_{k(-)} - \overline{K}_k H_k P_{k(-)}\\ &\quad - P_{k(-)} H_k^{\mathrm{T}} \overline{K}_k^{\mathrm{T}} + \overline{K}_k H_k P_{k(-)} H_k^{\mathrm{T}} \overline{K}_k^{\mathrm{T}} + \overline{K}_k R_k \overline{K}_k^{\mathrm{T}}\\ &= (I - \overline{K}_k H_k) P_{k(-)} - P_{k(-)} H_k^{\mathrm{T}} \overline{K}_k^{\mathrm{T}}\\ &\quad + \underbrace{\overline{K}_k (H_k P_{k(-)} H_k^{\mathrm{T}} + R_k)}_{P_{k(-)} H_k^{\mathrm{T}}} \overline{K}_k^{\mathrm{T}}\\ &= (I - \overline{K}_k H_k) P_{k(-)}, \end{aligned} \tag{5.24}$$

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,

$$\begin{aligned}P_{k(-)} &= \mathrm{E}\langle \tilde{x}_{k(-)} \tilde{x}_{k(-)}^{\mathrm{T}} \rangle,\\ \hat{x}_{k(-)} &= \Phi_{k-1} \hat{x}_{k-1(+)}, \end{aligned} \tag{5.25}$$

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 $\mathrm{E}\tilde{x}_{k-1}\ w_{k-1}^{\mathrm{T}} = 0$ to obtain the results

$$\begin{aligned} P_{k(-)} &\stackrel{\text{def}}{=} \mathrm{E}\langle \tilde{x}_{k(-)}\tilde{x}_{k(-)}^{\mathrm{T}}\rangle \\ &= \Phi_{k-1}\mathrm{E}\langle \tilde{x}_{k-1(+)}\tilde{x}_{k-1(+)}^{\mathrm{T}}\rangle\Phi_{k-1}^{\mathrm{T}} + \mathrm{E}\langle w_{k-1}\ w_{k-1}^{\mathrm{T}}\rangle \\ &= \Phi_{k-1}P_{k-1}^{(+)}\Phi_{k-1}^{\mathrm{T}} + Q_{k-1}, \end{aligned} \tag{5.26}$$

which gives the a priori value of the covariance matrix of estimation uncertainty as a function of the previous a posteriori value.

5.2.4 Kalman Gain from Gaussian Maximum Likelihood

The original derivation of the Kalman gain makes the fewest possible assumptions but requires the most mathematical rigor to obtain the most general result. The essential elements of that proof have been covered in the previous sections and Chapter 4.

An alternative derivation using the *linear Gaussian maximum-likelihood estimator* (*LGMLE* or *GMLE*) makes much more restrictive assumptions about the distribution of the state vector and measurements, but the resulting formula for the Kalman gain is the same. This derivation was introduced after some instructors observed symptoms resembling post traumatic stress disorder among students struggling with the more rigorous derivation.

In essence, this approach uses the mean μ_x and the information matrix $Y_{xx} = P_{xx}^{-1}$ as parameters for Gaussian distributions, then dismisses the Gaussian normalization factors to allow Y_{xx} to represent measurements that would render it singular.

The resulting functions are no longer probability functions, because their integrals are no longer 1 (one) and are not necessarily finite. They are called *Gaussian likelihood functions*, and they have properties similar to those of Gaussian probability distributions for joint and independent likelihoods. However, because integrals of likelihood functions are not necessarily defined, *expectation* can no longer be used to characterize likelihoods the way it characterizes probability measures. Least-mean-squared estimation error is not defined for likelihood functions, so it is replaced by *maximum likelihood* as a criterion for optimal estimation.

The means and information matrices of Gaussian likelihood functions are the parameters used in deriving the Kalman gain.

Example 5.1 (Combining Independent Gaussian Likelihoods) Consider the two-dimensional Gaussian likelihood function with a singular information matrix,

$$Y_a = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix},$$

in which case the 2D Gaussian likelihood function will have a shape like that shown in Figure 5.1(a), with the direction of "no information" shown by the double-ended arrow indicating the direction of the zero eigenvector of Y_{xx}.

This is a situation that could not be represented by a Gaussian probability density function, because its integral is not finite. The likelihood function, on the other hand, can represent the fact that there is no information in one direction. In essence, the eigenvalues of the information matrix represent *how much* information is available in the direction of the corresponding eigenvectors.

This also illustrates a case in which there is no unique location of the maximum of the likelihood function. It achieves its maximum along an entire infinite line.

The 2D Gaussian likelihood shown in Figure 5.1(b) represents the situation with information matrix

$$Y_b = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix},$$

in which case the direction of the zero eigenvector is orthogonal to that shown in Figure 5.1(a).

If these two likelihood functions are *independent,* in the sense that their joint likelihood is the point-wise product of the individual likelihoods, then their joint likelihood will be as illustrated in Figure 5.1(c). This resembles a 2D Gaussian likelihood function—and indeed it is. The formula for the Kalman gain can be derived by solving for the mean μ_c and information matrix Y_c of the likelihood shown in Figure 5.1(c) as a function of the means and information matrices of the likelihoods shown in Figure 5.1(a) and (b).

This alternative derivation of the formula for the Kalman gain matrix $\overline{K}_k$ uses the analogies shown in Figure 5.2 between the variable parameters of Kalman filtering ($\hat{x}$, P_{xx}), Gaussian probability densities (μ_x, P_{xx}), and Gaussian likelihood functions (μ_x, Y_{xx}).

5.2.4.1 Gaussian Maximum-Likelihood Estimation The theoretical basis for ML estimation was given its present form by Ronald A. Fisher (1890–1962) in the period 1912–1930 [23]. This work could draw upon on a body of theory going back more than a century [24–27].

An early application was for finding parameters of a distribution, given samples [26]. For Gaussian distributions, this is equivalent to finding the mean and covariance of a distribution. When Gaussian ML is formulated as a recursive estimator, this is analogous to the Kalman filtering problem if one equates the mean with the estimate, the covariance with the mean-squared estimation error, and samples with measurements. This analogy also provides a simple and direct derivation of the Kalman gain, resulting in the same formula derived by Kalman [28].

Gaussian Likelihoods and Log-Likelihoods The *Gaussian probability density function* for $x \in \mathcal{N}(\mu_x, P_{xx})$ is

$$p(x, \mu_x, P_{xx}) = \frac{1}{\sqrt{2\pi \det P_{xx}}} \exp\left(-\frac{1}{2}(x-\mu_x)^{\mathrm{T}} P_{xx}^{-1}(x-\mu_x)\right), \qquad (5.27)$$

where μ_x is the mean of X and P_{xx} is the covariance (second central moment).

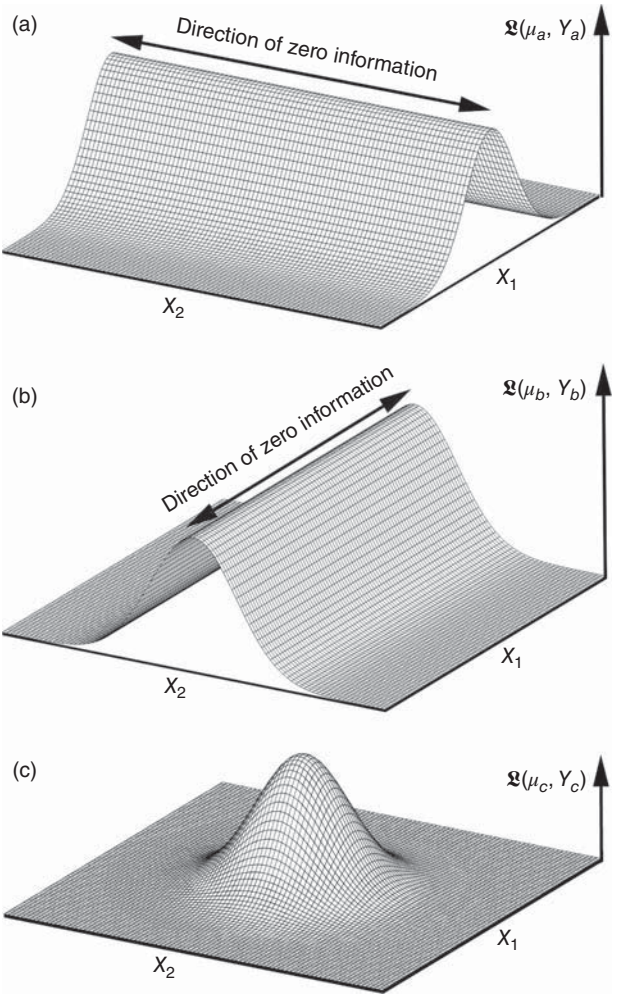


Figure 5.1 Products of Gaussian likelihoods.

The *Gaussian likelihood function* equivalent to Equation 5.27 would be of the same form

$$\mathcal{L}(x,\ \mu_x,\ Y_{xx}) = c\ \exp\ \left(-\frac{1}{2}(x-\mu_x)^{\mathrm{T}} Y_{xx}(x-\mu_x)\right), \tag{5.28}$$

except that is using the *information matrix* $Y_{xx} = P_{xx}^{-1}$ as a parameter. The scaling constant $c > 0$ plays no role in estimation, other than to allow such scaling to be ignored. The information matrix Y_{xx}, on the other hand, adds some modeling capabilities that are unavailable in probability modeling.

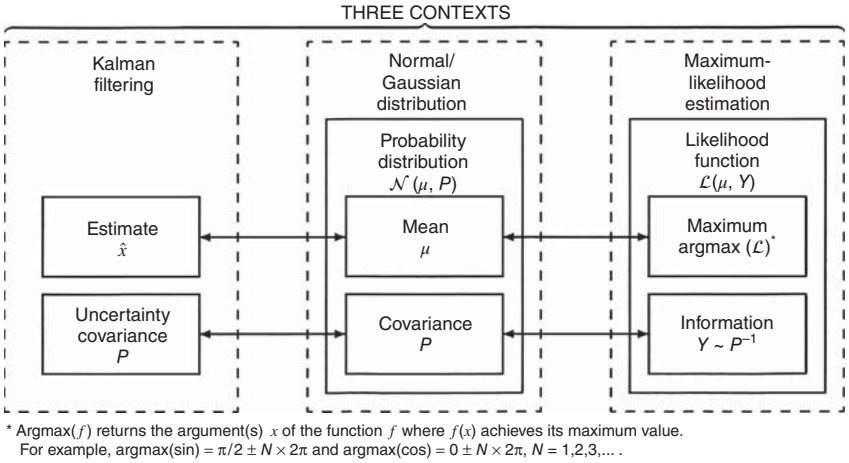


Figure 5.2 Analogous concepts in three different contexts.

Most of the useful work of Gaussian likelihoods is done using their logarithms, or *log-likelihoods*:

$$\log\ [\mathcal{L}(x,\ \mu_x,\ Y_{xx})] = \log(c) - \frac{1}{2}(x-\mu_x)^{\mathrm{T}} Y_{xx}(x-\mu_x). \tag{5.29}$$

Because the Gaussian likelihoods are always positive valued and the logarithm is a monotonically increasing function of positive numbers, maximizing the log-likelihood is equivalent to maximizing the likelihood.

5.2.4.2 Information Matrices All covariance matrices in Kalman filtering are symmetric and positive definite, because the variances of estimated quantities are never absolutely zero. Even fundamental constants of physics have associated error variances, and the value of π represented in finite-precision arithmetic has some roundoff error.

Consequently, all covariance matrices P_{xx} in Kalman filtering will have a matrix inverse $Y_{xx} = P_{xx}^{-1}$, the corresponding information matrix. In Kalman filtering, because P_{xx} is always symmetric and positive definite, the corresponding information matrix Y_{xx} will also be symmetric and positive definite. In fact, they have the same eigenvectors, and the corresponding eigenvalues of Y_{xx} will be the reciprocals of those of P_{xx}.

In ML estimation, however, the information matrices Y_{xx} are only symmetric and *nonnegative definite* (i.e., with zero eigenvalues possible) and, therefore, *not necessarily invertible.*

Using the information matrix in place of the covariance matrix in Gaussian likelihood functions allows us to model what estimation theorists would call "flat priors," a condition under which prior assumptions have no influence on the ultimate estimate. This cannot be done using covariance matrices, because it would require that some

eigenvalues be infinite. It can be done using information matrices by allowing them to have zero eigenvalues whose eigenvectors represent linear combinations of the state space in which there is zero information. For example, information matrices can be used to represent the information in a measurement, and the dimension of which may be less than the dimension of the state vector.

Example 5.2 (GNSS Pseudorange Measurement) Global navigation satellite navigation systems (GNSS) use accurate onboard clocks and precise time stamping of transmitted signals, so that receivers with accurate synchronized clocks can determine the propagation time delay, and from that an estimate of the distance the signal has traveled between the transmitting antenna and the receiving antenna. The estimate ρ is called a *pseudorange* because it also includes errors due to the receiver clock and atmospheric propagation delays.

Each pseudorange is then a one-dimensional measurement of the location of the receiver antenna relative to the satellite antenna, the location of which is known. If u is a unit vector in the direction from the transmitting antenna to the receiver antenna, then the partial derivative of the pseudorange measurement with respect to receiver antenna location x (a 3-vector) will be the 1×3 vector

$$\frac{\partial \rho}{\partial x} = u^{\mathrm{T}}.$$

The information that this measurement adds to the navigation solution (3D position) is represented by the 3×3 *information matrix*

$$Y_{xx} = uR^{-1}u^{\mathrm{T}},$$

where R is the mean-squared "sensor noise," the error in ρ due to signal processing noise, etc. This information matrix has rank 1, with one nonzero eigenvector u and two zero eigenvectors orthogonal to it.

Singular Value Decompositions of Information Matrices Singular value decomposition (svd) is used as a diagnostic tool for characterizing certain properties of matrices, and it offers insights into how information matrices behave.

The svd of an arbitrary $m \times n$ real matrix M is a factoring in the form $M = LDR$, where L is an $m \times m$ unitary matrix, R is an $n \times n$ unitary matrix, and D is an $m \times n$ matrix with nonnegative values down its main diagonal—with zeros elsewhere. The columns of L are called the *left eigenvectors* of M, the rows of R are called the *right eigenvectors,* and the main diagonal elements of D are called the *singular values* of M.

Because an information matrix Y_{xx} is symmetric and nonnegative definite, its svd will have the same left and right eigenvectors, which will be real vectors. In that

case, $L = R^{\mathrm{T}} = V$ and its svd can be expressed in terms of its eigenvalues λ_i (all nonnegative) and corresponding real eigenvectors e_i as

$$Y_{xx} = \sum_i \lambda_i e_i e_i^{\mathrm{T}} \tag{5.30}$$

$$= V \operatorname{diag}(\lambda) V^{\mathrm{T}} \tag{5.31}$$

$$V = \begin{bmatrix} e_1 & e_2 & e_3 & \cdots & e_n \end{bmatrix} \tag{5.32}$$

$$\operatorname{diag}(\lambda) = \begin{bmatrix} \lambda_1 & 0 & 0 & \cdots & 0 \\ 0 & \lambda_2 & 0 & \cdots & 0 \\ 0 & 0 & \lambda_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \lambda_n \end{bmatrix}, \tag{5.33}$$

The eigenvalues λ_i in the svds have the largest eigenvalue first, followed by the rest in order of decreasing value:

$$\lambda_1 \geq \lambda_2 \geq \lambda_2 \geq \cdots \geq \lambda_n.$$

Consequently, if Y_{xx} has rank $n - r$, the last r eigenvalues will be zero.

Example 5.3 (Singular Value Decompositions of 2×2 Information Matrices) The 2×2 information matrices used in Example 5.1 and their corresponding svds and eigenvalue–eigenvector decompositions can be summarized as

	Y	=	L	×	D	×	R	λ_1	e_1	λ_2	e_2
(a)	$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$	=	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$		$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$		$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	1	$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$	0	$\begin{bmatrix} 0 \\ 1 \end{bmatrix}$
(b)	$\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$	=	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$		$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$		$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$	1	$\begin{bmatrix} 0 \\ 1 \end{bmatrix}$	0	$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$
(c)	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	=	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$		$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$		$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	1	$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$	1	$\begin{bmatrix} 0 \\ 1 \end{bmatrix}$

Moore–Penrose generalized matrix inverse The *Moore–Penrose generalized inverse* of Y_{xx} can be defined in terms of its svd as

$$Y_{xx}^{\dagger} = \sum_{\lambda_i \neq 0} \lambda_i^{-1} e_i e_i^{\mathrm{T}}, \tag{5.34}$$

which is always symmetric and nonnegative definite and of the same rank as Y_{xx}.

5.2.4.3 Formulas for Joint Independent Likelihoods Two probability distributions are called *statistically independent* if and only if their joint probability is the product of the individual probabilities. The same is true for likelihoods.

Following the notation of Figure 5.1, let the joint likelihood function $\mathcal{L}_c(x,\ \mu_c,\ Y_c)$ of two independent Gaussian likelihoods $\mathcal{L}_a(x,\ \mu_a,\ Y_a)$ and $\mathcal{L}_b(x,\ \mu_b,\ Y_b)$ be represented by the product of the two likelihood functions:

$$c_c \exp\left(-\frac{1}{2}\left(x-\mu_c\right)^{\mathrm{T}} Y_c (x-\mu_c)\right)$$

$$= \mathcal{L}_c(x, \mu_c, Y_c) \tag{5.35}$$

$$= \mathcal{L}_a(x, \mu_a, Y_a) \times \mathcal{L}_b(x, \mu_b, Y_b) \tag{5.36}$$

$$= c_a \exp\left(-\frac{1}{2}\left(x-\mu_a\right)^{\mathrm{T}} Y_a (x-\mu_a)\right) \times c_b \exp\left(-\frac{1}{2}\left(x-\mu_b\right)^{\mathrm{T}} Y_b (x-\mu_b)\right) \tag{5.37}$$

$$= c_a c_b \exp\left(-\frac{1}{2}\left(x-\mu_a\right)^{\mathrm{T}} Y_a (x-\mu_a) - \frac{1}{2}(x-\mu_b)^{\mathrm{T}} Y_b (x-\mu_b)\right). \tag{5.38}$$

Taking the logarithm of both sides and differentiating once and twice with respect to x will yield the following sequence of equations:

$$\log\ (c_c) - \frac{1}{2}(x-\mu_c)^{\mathrm{T}} Y_c (x-\mu_c)$$

$$= \log\ (c_a) + \log\ (c_b) - \frac{1}{2}(x-\mu_a)^{\mathrm{T}} Y_a (x-\mu_a)$$

$$-\frac{1}{2}(x-\mu_b)^{\mathrm{T}} Y_b (x-\mu_b) \tag{5.39}$$

$$Y_c(x-\mu_c) = Y_a(x-\mu_a) + Y_b(x-\mu_b) \tag{5.40}$$

$$Y_c = Y_a + Y_b, \tag{5.41}$$

the last line of which says that *information is additive.* Setting $x = 0$ in the next-to-last line yields the equation

$$Y_c \mu_c = Y_a \mu_a + Y_b \mu_b. \tag{5.42}$$

Equations 5.41 and 5.42, with appropriate substitution of Kalman filtering variables, are all that is needed to solve for the Kalman gain.

5.2.4.4 Solving for the Kalman Gain

Substitutions The following substitutions will be made in Equations 5.41 and 5.42:

$$\left.\begin{array}{rcll} \mu_a & = & \hat{x}_{k(-)} & \text{a priori estimate} \\ Y_a & = & P_{k(-)}^{-1} & \text{a priori information} \\ \mu_b & = & H_k^{\dagger} z_k & \text{measurement mean} \\ Y_b & = & H_k^{\mathrm{T}} R_k^{-1} H_k & \text{measurement information} \\ \mu_c & = & \hat{x}_{k(+)} & \text{a posteriori estimate} \\ Y_c & = & P_{k(+)}^{-1} & \text{a posteriori information,} \end{array}\right\} \tag{5.43}$$

where $z_k = H_k \hat{x}_{k(-)} + v_k$ is the measurement, H_k is the measurement sensitivity matrix, v_k is the noise on the measurement, and R_k is the covariance of v_k.

Solving for the Covariance Update With the substitutions of Equation 5.43, Equation 5.41 becomes

$$P_{k(+)}^{-1} = P_{k(-)}^{-1} + H_k^{\mathrm{T}} R_k^{-1} H_k, \tag{5.44}$$

where $P_{k(-)}^{-1}$ is the a priori state information and $H_k^{\mathrm{T}} R_k^{-1} H_k = Y_b$ is the information in the kth measurement z_k.

An Inverse Matrix Modification Formula We can use the general formula for the inverse of a matrix sum due to Duncan [29] (among others)

$$(A^{-1} + BC^{-1}D)^{-1} = A - AB(C + DAB)^{-1}DA. \tag{5.45}$$

More Substitutions With the substitutions

$$\left.\begin{array}{rcll} A^{-1} & = & Y_a, & \text{the a priori information matrix for } \hat{x} \\ A & = & P_{k(-)}, & \text{the a priori covariance matrix for } \hat{x} \\ B & = & H_k^{\mathrm{T}}, & \text{transpose of the measurement sensitivity matrix} \\ C & = & R_k, & \text{covariance of measurement noise } v_k \\ D & = & H_k, & \text{the measurement sensitivity matrix,} \end{array}\right\} \tag{5.46}$$

Equation 5.45 becomes

$$\left.\begin{array}{rcll} P_{k(+)} & = & Y_c^{-1} & \\ & = & \left(Y_a + H_k^{\mathrm{T}} R_k^{-1} H_k\right)^{-1} & \text{(Eq. 5.44)} \\ & = & Y_a^{-1} - Y_a^{-1} H_k^{\mathrm{T}} (H_k Y_a^{-1} H_k^{\mathrm{T}} + R_k)^{-1} H_k Y_A^{-1} & \text{(Eq. 5.45)} \\ & = & P_{k(-)} - \underbrace{P_{k(-)} H_k{}^{\mathrm{T}} (H_k P_{k(-)} H_k{}^{\mathrm{T}} + R_k)^{-1}}_{\overline{K}_k} H_k P_{k(-)}, & \text{(Eq. 5.43)} \end{array}\right\} \tag{5.47}$$

where the expression labeled "$\overline{K}_k$" will also be found in the derivation of the update of the estimate in the following.

Solving for the Estimate Update Equation 5.42 with substitutions from Equation 5.43 will have the form

$$
\begin{aligned}
\hat{x}_{k(+)} &= \mu_c && \text{(Eq. 5.43)}\\
&= Y_c^{-1}(Y_a\mu_a + Y_b\mu_b) && \text{(Eq. 5.42)}\\
&= P_{k(+)}\left[P_{k(-)}^{-1}\hat{x}_{k(-)} + H_k^{\mathrm{T}}R_k^{-1}H_kH_k^{\dagger}z_k\right] && \text{(Eq. 5.43)}\\
&= [P_{k(-)} - P_{k(-)}H_k^{\mathrm{T}}(H_kP_{k(-)}H_k^{\mathrm{T}} + R_k)^{-1}H_kP_{k(-)}] && \text{(Eq. 5.47)}\\
&\quad \times [P_{k(-)}^{-1}\hat{x}_{k(-)} + H_k^{\mathrm{T}}R_k^{-1}H_kH_k^{\dagger}z]\\
&= [I - P_{k(-)}H_k^{\mathrm{T}}(H_kP_{k(-)}H_k^{\mathrm{T}} + R_k)^{-1}H_k]\\
&\quad \times [\hat{x}_{k(-)} + P_{k(-)}H_k^{\mathrm{T}}R_k^{-1}H_kH_k^{\dagger}z]\\
&= \hat{x}_{k(-)} + P_{k(-)}H_k^{\mathrm{T}}(H_kP_{k(-)}H_k^{\mathrm{T}} + R_k)^{-1}\\
&\quad \times \{[(H_kP_{k(-)}H_k^{\mathrm{T}} + R_k)R_k^{-1} - H_kP_{k(-)}\,H_k^{\mathrm{T}}R_k^{-1}]z_k - H_k\hat{x}_{k(-)}\}\\
&= \hat{x}_{k(-)} + P_{k(-)}H_k^{\mathrm{T}}(H_kP_{k(-)}H_k^{\mathrm{T}} + R_k)^{-1}\\
&\quad \times \{[H_kP_{k(-)}H_k^{\mathrm{T}}R_k{}^{-1} + I - H_kP_{k(-)}H_k^{\mathrm{T}}R_k{}^{-1}]z_k - H_k\hat{x}_{k(-)}\}\\
&= \hat{x}_{k(-)} + \underbrace{\{P_{k(-)}H_k^{\mathrm{T}}(H_kP_{k(-)}H_k^{\mathrm{T}} + R_k)^{-1}\}}_{\overline{K}_k}[z_k - H_k\hat{x}_{k(-)}].
\end{aligned}
$$

The Kalman Gain The last equation of the above can now be rewritten as two equations,

$$\hat{x}_{k(+)} = \hat{x}_{k(-)} + \overline{K}_k[z_k - H_k\hat{x}_{k(-)}] \tag{5.48}$$

$$\overline{K}_k = P_{k(-)}H_k{}^{\mathrm{T}}(H_kP_{k(-)}H_k^{\mathrm{T}} + R_k)^{-1}, \tag{5.49}$$

the first of which models the estimation update in feedback form, with the Kalman gain given by the second equation.

5.2.4.5 Other Argmax Estimators The linear Gaussian ML estimator was used above as an alternative derivation of the Kalman gain formula. In this case, the resulting estimate (argmax of the likelihood function) is the mode, mean, and median of the underlying Gaussian distribution.

There is an alternative class of estimators called *maximum a posteriori probability* (MAP) estimators which use Bayes Rule to compute the argmax of the a posteriori probability density function to select the value of the variable to be estimated at which its probability density is greatest (maximum mode). These estimators are applicable to a more general class of problems (including non-Gaussian and nonlinear) than the Kalman filter, but they tend to have computational complexities that would eliminate them from consideration for real-time practical implementations as filters. They are used for some nonlinear and nonreal-time applications, however. (See, e.g., Bain and Crisan [30] or Crassidis and Jenkins [31].)

5.2.5 Kalman Gain from Recursive Linear LMS Estimator

This is another low stress derivation of the Kalman gain, starting from the linear LMS estimator in recursive form. It makes fewer assumptions than the derivation from Gaussian ML estimation and uses just a bit of matrix arithmetic.

5.2.5.1 Linear Least Mean Squares If the error vector v in the least-squares problem has a known covariance R,

$$z = Hx + v \tag{5.50}$$

$$R \overset{\text{def}}{=} \underset{v}{\mathrm{E}}\langle vv^{\mathrm{T}}\rangle \tag{5.51}$$

with svd

$$R = U_R D_R U_R^{\mathrm{T}} \tag{5.52}$$

as a product of orthogonal (U) and diagonal (D) matrices, then its inverse

$$R^{-1} = U_R D_R^{-1} U_R^{\mathrm{T}} \tag{5.53}$$

has a symmetric matrix square root

$$S_R \overset{\text{def}}{=} U_R D_R^{-1/2} U_R^{\mathrm{T}} \tag{5.54}$$

$$S_R^2 = U_R D_R^{-1/2} \underbrace{U_R^{\mathrm{T}} U_R}_{I} D_R^{-1/2} U_R^{\mathrm{T}} \tag{5.55}$$

$$= U_R D_R^{-1/2} D_R^{-1/2} U_R^{\mathrm{T}} \tag{5.56}$$

$$= U_R D_R^{-1} U_R^{\mathrm{T}} \tag{5.57}$$

$$= R^{-1}. \tag{5.58}$$

Consequently, the rescaled least-squares problem

$$\underbrace{S_R z}_{z^\star} = \underbrace{S_R H}_{H^\star} x + \underbrace{S_R v}_{v^\star} \tag{5.59}$$

has error covariance

$$\begin{aligned}
R^\star &\stackrel{\text{def}}{=} \mathop{\mathrm{E}}_{v}\langle v^\star v^{\star\mathrm{T}}\rangle && (5.60)\\
&= \mathop{\mathrm{E}}_{v}\langle S_R v v^{\mathrm{T}} S_R^{\mathrm{T}}\rangle && (5.61)\\
&= S_R \mathop{\mathrm{E}}_{v}\langle v v^{\mathrm{T}}\rangle S_R^{\mathrm{T}} && (5.62)\\
&= (U_R D_R^{-1/2} U_R^{\mathrm{T}}) R (U_R D_R^{-1/2} U_R^{\mathrm{T}}) && (5.63)\\
&= (U_R D_R^{-1/2} U_R^{\mathrm{T}})(U_R D_R U_R^{\mathrm{T}})(U_R D_R^{-1/2} U_R^{\mathrm{T}}) && (5.64)\\
&= U_R D_R^{-1/2} \underbrace{U_R^{\mathrm{T}} U_R}_{I} D_R \underbrace{U_R^{\mathrm{T}} U_R}_{I} D_R^{-1/2} U_R^{\mathrm{T}} && (5.65)\\
&= U_R \underbrace{D_R^{-1/2} D_R D_R^{-1/2}}_{I} U_R^{\mathrm{T}} && (5.66)\\
&= U_R U_R^{\mathrm{T}} && (5.67)\\
&= I. && (5.68)
\end{aligned}$$

That is, the rescaled error covariance is an identity matrix, meaning that the individual scalar errors are uncorrelated and all have the same variance (1).

In that case, the rescaled least-squares problem (Equation 5.59)

$$S_R z = S_R H x + v^\star \tag{5.69}$$

has solution

$$\begin{aligned}
\hat{x} &= [H^{\star T} H^\star]^{-1} H^{\star T} z^\star && (5.70)\\
&= [(S_R H)^{\mathrm{T}} (S_R H)]^{-1} (S_R H)^{\mathrm{T}} S_R z && (5.71)\\
&= [(S_R H)^{\mathrm{T}} (S_R H)]^{-1} (S_R H)^{\mathrm{T}} S_R z && (5.72)\\
&= [H^{\mathrm{T}} S_R^{\mathrm{T}} S_R H]^{-1} H^{\mathrm{T}} S_R^{\mathrm{T}} S_R z && (5.73)\\
&= [H^{\mathrm{T}} R^{-1} H]^{-1} H^{\mathrm{T}} R^{-1} z, && (5.74)
\end{aligned}$$

provided that R and the *information matrix* (the analog of the gramian in least-squares estimation)

$$Y \stackrel{\text{def}}{=} H^{\mathrm{T}} R^{-1} H \tag{5.75}$$

are nonsingular, in which case

$$Y^{-1} = P, \tag{5.76}$$

the covariance of estimation uncertainty with all errors independent and having equal variance. That $\hat{x}$ is the LMS estimate.

5.2.5.2 Recursive LMS If the observation vector z can be partitioned into subvectors

$$\mathcal{Z}_k = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ \vdots \\ z_k \end{bmatrix} \tag{5.77}$$

such that the associated additive noise subvectors are uncorrelated,

$$\underset{v}{\mathrm{E}}\langle v_i v_j^{\mathrm{T}} \rangle = \begin{cases} 0, & i \neq j \\ R_i, & i = j, \end{cases} \tag{5.78}$$

then the matrices H and R can be similarly partitioned as

$$\mathcal{H}_k = \begin{bmatrix} H_1 \\ H_2 \\ H_3 \\ \vdots \\ H_k \end{bmatrix} \text{ and } \mathcal{R}_k = \begin{bmatrix} R_1 & 0 & 0 & \cdots & 0 \\ 0 & R_2 & 0 & \cdots & 0 \\ 0 & 0 & R_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & R_k \end{bmatrix}$$

and the associated information and covariance matrices

$$Y_k = \sum_{\ell=1}^{k} H_\ell^{\mathrm{T}} R_\ell^{-1} H_\ell \tag{5.79}$$

$$= Y_{k-1} + H_k^{\mathrm{T}} R_k^{-1} H_k \tag{5.80}$$

$$P_k = Y_k^{-1} \tag{5.81}$$

$$= \{P_{k-1}^{-1} + H_k^{\mathrm{T}} R_k^{-1} H_k\}^{-1} \tag{5.82}$$

to which one can apply the general "modified matrix" inversion formula 5.45 with

$$A = P_{k-1}^{-1} \tag{5.83}$$

$$B = H_k^{\mathrm{T}} \tag{5.84}$$

$$C = -R_k \tag{5.85}$$

$$D = H_k \tag{5.86}$$

to obtain

$$P_k = P_{k-1} - \underbrace{P_{k-1}H_k^{\mathrm{T}}[R_k + H_k P_{k-1} H_k^{\mathrm{T}}]^{-1}}_{\overline{K}_k} H_k P_{k-1}, \tag{5.87}$$

the formula for covariance matrix measurement updates of the Kalman filter, and in which you might recognize the underbraced expression as our quarry:

$$\overline{K}_k = P_{k-1}H_k^{\mathrm{T}}[R_k + H_k P_{k-1} H_k^{\mathrm{T}}]^{-1}.$$

However, it remains to be proven that this value provides the optimal linear gain for recursive LMSs estimation.

Recursive LMS Estimate The corresponding recursive update of the estimate will be

$$\hat{x}_k = P_k \left\{ \sum_{\ell=1}^{k} H_\ell^{\mathrm{T}} R_\ell^{-1} z_\ell \right\} \tag{5.88}$$

$$= \{P_{k-1} - \overline{K}_k H_k P_{k-1}\} \left\{ \sum_{\ell=1}^{k-1} H_\ell^{\mathrm{T}} R_\ell^{-1} z_\ell + H_k^{\mathrm{T}} R_k^{-1} z_k \right\} \tag{5.89}$$

$$= \{I - \overline{K}_k H_k\} \left\{ P_{k-1} \sum_{\ell=1}^{k-1} H_\ell^{\mathrm{T}} R_\ell^{-1} z_\ell + P_{k-1} H_k^{\mathrm{T}} R_k^{-1} z_k \right\} \tag{5.90}$$

$$= \{I - \overline{K}_k H_k\} \{\hat{x}_{k-1} + P_{k-1} H_k^{\mathrm{T}} R_k^{-1} z_k\} \tag{5.91}$$

$$= x_{k-1} - \overline{K}_k H_k x_{k-1} + \{I - \overline{K}_k H_k\} P_{k-1} H_k^{\mathrm{T}} R_k^{-1} z_k \tag{5.92}$$

$$= x_{k-1} - \overline{K}_k H_k x_{k-1} + \{P_{k-1} H_k^{\mathrm{T}} R_k^{-1} - \overline{K}_k H_k P_{k-1} H_k^{\mathrm{T}} R_k^{-1}\} z_k \tag{5.93}$$

$$= x_{k-1} - \overline{K}_k H_k x_{k-1} + \mathcal{X}_k z_k, \tag{5.94}$$

where

$$\mathcal{X}_k \stackrel{\text{def}}{=} P_{k-1} H_k^{\mathrm{T}} \underbrace{\{I - [R_k + H_k P_{k-1} H_k^{\mathrm{T}}]^{-1} H_k P_{k-1} H_k^{\mathrm{T}}\} R_k^{-1}}_{\mathcal{Y}_k} \tag{5.95}$$

$$\mathcal{Y}_k \stackrel{\text{def}}{=} \{I - [R_k + H_k P_{k-1} H_k^{\mathrm{T}}]^{-1} H_k P_{k-1} H_k^{\mathrm{T}}\} R_k^{-1} \tag{5.96}$$

$$[R_k + H_k P_{k-1} H_k^{\mathrm{T}}]\mathcal{Y}_k = \{R_k + H_k P_{k-1} H_k^{\mathrm{T}} - H_k P_{k-1} H_k^{\mathrm{T}}\} R_k^{-1} \tag{5.97}$$

$$= \{R_k\} R_k^{-1} \tag{5.98}$$

$$= I \tag{5.99}$$

$$\mathcal{Y}_k = [R_k + H_k P_{k-1} H_k^{\mathrm{T}}]^{-1} \tag{5.100}$$

$$\mathcal{K}_k = P_{k-1}H_k^{\mathrm{T}}\mathcal{Y}_k \tag{5.101}$$

$$= P_{k-1}H_k^{\mathrm{T}}[R_k + H_k P_{k-1}H_k^{\mathrm{T}}]^{-1} \tag{5.102}$$

$$\stackrel{\text{def}}{=} \overline{K}_k, \tag{5.103}$$

so that Equation 5.94 becomes

$$\hat{x}_k = \hat{x}_{k-1} - \overline{K}_k H_k \hat{x}_{k-1} + \overline{K}_k z_k \tag{5.104}$$

$$= \hat{x}_{k-1} + \overline{K}_k(z_k - H_k\hat{x}_{k-1}). \tag{5.105}$$

That is, the formula for the recursive LMS estimate in Equation 5.105 is exactly the same as the Kalman measurement update formula, and with the same Kalman gain, as given by Equation 5.88.

This completes the derivation of the Kalman gain from the recursive LMS estimator.

5.2.6 Summary of Equations for the Discrete-Time Kalman Estimator

The equations derived in the previous section are summarized in Table 5.3. In this formulation of the filter equations, G has been combined with the plant covariance by multiplying G_{k-1} and G_{k-1}^{T}, for example,

$$Q_{k-1} = G_{k-1}\mathrm{E}\langle w_{k-1}w_{K-1}^{\mathrm{T}}\rangle G_{k-1}^{\mathrm{T}}$$
$$= G_{k-1}\overline{Q}_{k-1}G_{k-1}^{\mathrm{T}}.$$

The relation of the filter to the system is illustrated in the block diagram of Figure 5.3.

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}.

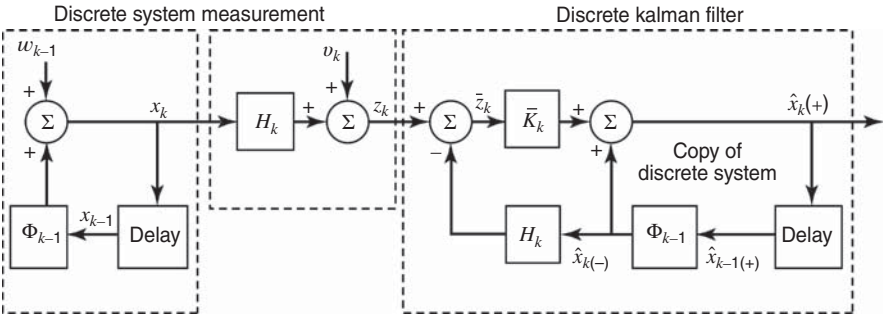


Figure 5.3 Block diagram of system, measurement model, and discrete-time Kalman filter.

TABLE 5.3 Discrete-Time Kalman Filter Equations

System dynamic model	$x_k = \Phi_{k-1}x_{k-1} + w_{k-1}$ $w_k \sim \mathcal{N}(0, Q_k)$
Measurement model	$z_k = H_k x_k + v_k$ $v_k \sim \mathcal{N}(0, R_k)$
Initial conditions	$\mathrm{E}\langle x_0 \rangle = \hat{x}_0$ $\mathrm{E}\langle \tilde{x}_0 \tilde{x}_0^{\mathrm{T}} \rangle = P_0$
Independence assumption	$\mathrm{E}\langle w_k v_j^{\mathrm{T}} \rangle = 0$ for all k and j
State estimate extrapolation (Equation 5.25)	$\hat{x}_{k(-)} = \Phi_{k-1}\hat{x}_{(k-1)(+)}$
Error covariance extrapolation (Equation 5.26)	$P_{k(-)} = \Phi_{k-1}P_{(k-1)(+)}\Phi_{k-1}^{\mathrm{T}} + Q_{k-1}$
State estimate observational update (Equation 5.21)	$\hat{x}_{k(+)} = \hat{x}_{k(-)} + \overline{K}_k[z_k - H_k\hat{x}_{k(-)}]$
Error covariance update (Equation 5.24)	$P_{k(+)} = [I - \overline{K}_k H_k]P_{k(-)}$
Kalman gain matrix (Equation 5.19)	$\overline{K}_k = P_{k(-)}H_k^{\mathrm{T}}[H_k P_{k(-)}H_k^{\mathrm{T}} + R_k]^{-1}$

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.

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 5.11.

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 5.23:

$$P_{k(+)} = [I - \overline{K}_k H_k]P_{k(-)}[I - \overline{K}_k H_k]^{\mathrm{T}} + \overline{K}_k R_k \overline{K}_k^{\mathrm{T}}.$$

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.

[4] After Bucy and Joseph [6].

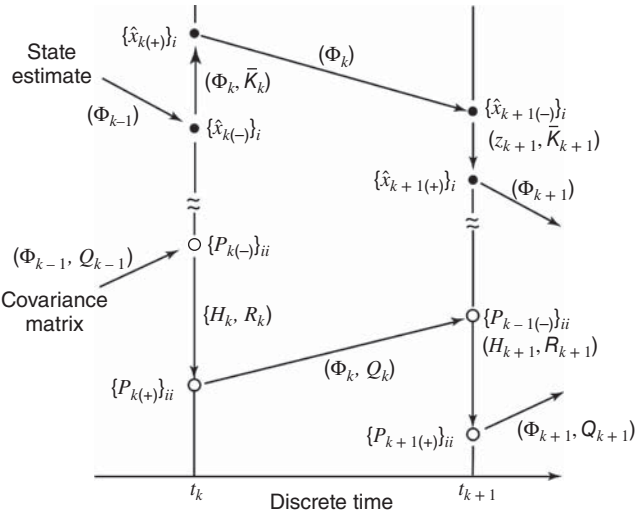


Figure 5.4 Representative sequence of values of filter variables in discrete time.

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 5.4 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 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.

Typical behavior of successive values of the diagonal elements of P_k, the covariance matrix of estimation uncertainty, is illustrated in Figure 5.5, which displays the "sawtooth" pattern between the a priori ($P_{k(-)}$) and a posteriori ($P_{k(+)}$) values of P_k. As a rule, for the same value of k, the diagonal values of $P_{k(+)}$ tend to be less than or equal to those of $P_{k(-)}$.

The filter would be said to *converge* if $P \to 0$ as $k \to \infty$, but this will not happen unless $Q = 0$ or $\Phi = 0$. An example of convergence with $Q = 0$ is shown in Figure 5.6, in which case the true state variable is a constant ($\Phi = 1$) and convergence is exponential. The state variable in this example is a scalar and the $\pm 1\sigma$ values bracketing the estimate are calculated from the square root of P. The separation of the

[5]Some of the alternative forms for computing $\overline{K}_k$ and $P_{k(+)}$ can be found in Jazwinski [10], Kailath [11], and Sorenson [32].

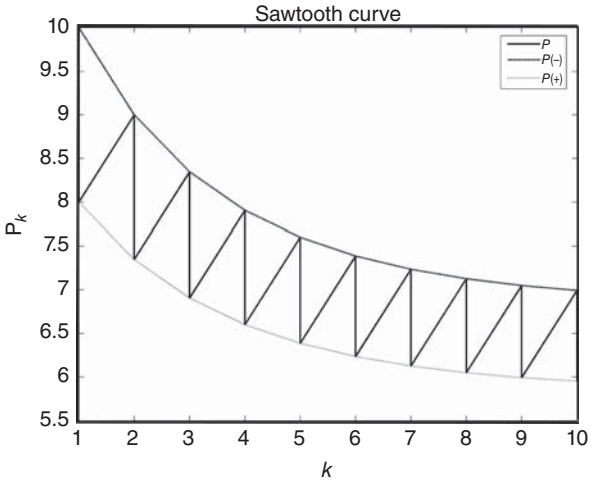


Figure 5.5 Sawtooth pattern of P_k diagonal values.

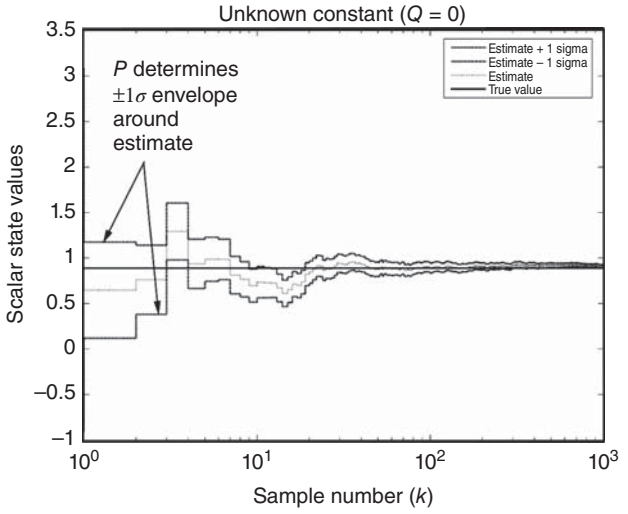


Figure 5.6 Convergence of P_k.

$\pm 1\sigma$ values is (slowly) converging to zero, which indicates that the filter is, indeed, converging.

In the more general case, P may still converge to some nonzero steady-state value.

Example 5.4 (Discrete-Time Implementation with Numerical Values) Let the scalar system dynamics and observations be given by the following equations:

$$x_k = x_{k-1} + w_{k-1}, \quad z_k = x_k + v_k,$$

$$\mathrm{E}\langle v_k \rangle = \mathrm{E}\langle w_k \rangle = 0,$$

$$\mathrm{E}\langle v_{k_1} v_{k_2} \rangle = 2\Delta(k_2 - k_1), \mathrm{E}\langle w_{k_1} w_{k_2} \rangle = \Delta(k_2 - k_1),$$

$$z_1 = 2, z_2 = 3,$$

$$\mathrm{E}\langle x(0) \rangle = \hat{x}_0 = 1,$$

$$\mathrm{E}\langle [x(0) - \hat{x}_0]^2 \rangle = P_0 = 10.$$

The objective is to find $\hat{x}_3$ and the steady-state covariance matrix P_∞. One can use the equations in Table 5.3 with

$$\Phi = 1 = H, Q = 1, R = 2,$$

for which

$$\boxed{P_{k(-)} = P_{(k-1)(+)} + 1},$$

$$\bar{K}_k = \frac{P_{k(-)}}{P_{k(-)} + 2} = \frac{P_{(k-1)(+)} + 1}{P_{(k-1)(+)} + 3},$$

$$P_{k(+)} = \left[1 - \frac{P_{(k-1)(+)} + 1}{P_{(k-1)(+)} + 3}\right](P_{(k-1)(+)} + 1),$$

$$\boxed{P_{k(+)} = \frac{2(P_{(k-1)(+)} + 1)}{P_{(k-1)(+)} + 3}},$$

$$\boxed{\hat{x}_{k(+)} = \hat{x}_{(k-1)(+)} + \bar{K}_k(z_k - \hat{x}_{(k-1)(+)})}.$$

Let

$$P_{k(+)} = P_{(k-1)(+)} = P \text{ (steady-state covariance)},$$

$$P = \frac{2(P+1)}{P+3},$$

$$P^2 + P - 2 = 0,$$

$$P = 1, \text{ positive-definite solution.}$$

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	$\frac{35}{13}$	$\frac{70}{61}$	$\frac{35}{61}$	$\frac{153}{61}$

5.2.7 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 7 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 5.5 (Serial Processing of Vector Measurements) Consider the measurement update problem with

$$\hat{x}_{k-} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \quad P_{k-} = \begin{bmatrix} 4 & 1 \\ 1 & 9 \end{bmatrix}, \quad H_k = \begin{bmatrix} 0 & 2 \\ 3 & 0 \end{bmatrix}, \quad R_k = \begin{bmatrix} 1 & 0 \\ 0 & 4 \end{bmatrix}, \quad z_k = \begin{bmatrix} 3 \\ 4 \end{bmatrix}.$$

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

$$R_1 = 1, \quad R_2 = 4,$$

and measurement sensitivity matrices

$$H_1 = \begin{bmatrix} 0 & 2 \end{bmatrix}, \quad H_2 = \begin{bmatrix} 3 & 0 \end{bmatrix}.$$

Table 5.4 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 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 resemblance to the two columns of Kalman gain matrix for vector-valued measurements.

TABLE 5.4 Calculations for Example 5.5

Implementation Equations	Vector Measurements	Scalar Measurements	
		First Measurement	Second Measurement
$\hat{z} = H\hat{x}_{k(-)}$	$\begin{bmatrix}4\\3\end{bmatrix}$	4	105/37*
$\tilde{z} = z - \hat{z}$	$\begin{bmatrix}-1\\1\end{bmatrix}$	−1	43/37
$[HP]$	$\begin{bmatrix}2 & 18\\12 & 3\end{bmatrix}$	$[2 \quad 18]$	$\begin{bmatrix}\frac{432}{37} & \frac{3}{37}\end{bmatrix}$*
$[[HP]H^{\mathrm{T}} + R]$	$\begin{bmatrix}37 & 6\\6 & 40\end{bmatrix}$	37	1444/37
$[[HP]H^{\mathrm{T}} + R]^{-1}$	$\begin{bmatrix}\frac{10}{361} & -\frac{3}{722}\\-\frac{3}{722} & \frac{37}{1444}\end{bmatrix}$	1/37	37/1444
$\overline{K} = (HP)^{\mathrm{T}}[HPH^{\mathrm{T}} + R]^{-1}$	$\begin{bmatrix}\frac{2}{361} & \frac{108}{361}\\\frac{351}{722} & \frac{3}{1444}\end{bmatrix}$	$\begin{bmatrix}\frac{2}{37}\\\frac{18}{37}\end{bmatrix}$	$\begin{bmatrix}\frac{108}{361}\\\frac{3}{1444}\end{bmatrix}$
$\hat{x}_{k(+)} = \hat{x}_{k(-)} + \overline{K}\tilde{z}$	$\begin{bmatrix}\frac{467}{361}\\\frac{2189}{1444}\end{bmatrix}$	†	$\begin{bmatrix}\frac{467}{361}\\\frac{2189}{1444}\end{bmatrix}$
$\hat{x}_{k(+)/2}$†		$\begin{bmatrix}\frac{35}{37}\\\frac{56}{37}\end{bmatrix}$	
$P_{k(+)} = P_{k(-)} - \overline{K}[HP]$	$\begin{bmatrix}\frac{144}{361} & \frac{1}{361}\\\frac{1}{361} & \frac{351}{1444}\end{bmatrix}$	†	$\begin{bmatrix}\frac{144}{361} & \frac{1}{361}\\\frac{1}{361} & \frac{351}{1444}\end{bmatrix}$
$P_{k(+)/2}$†		$\begin{bmatrix}\frac{144}{37} & 1/37\\1/37 & \frac{9}{37}\end{bmatrix}$	

*Note that $\hat{z}$ on the second pass is based on $\hat{x}_{k(+)/2}$ from the first pass, not $\hat{x}_{k(-)}$, and P on the second pass is $P_{k(+)/2}$ 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 multi-pass processing of measurement vector components increases with vector length.

5.2.8 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 of 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.

5.3 KALMAN–BUCY FILTER

Kalman and Bucy published the continuous-time equivalent of the Kalman filter [33] shortly after the Kalman filter first appeared [28]. It provides a good heuristic model for engineers who prefer to "think continuously, but act discretely." These are people who are more comfortable thinking about dynamics in continuous time, in terms of derivatives. The Kalman–Bucy filter allows them to develop a model in continuous time until they feel confident with it, then switch over to the equivalent Kalman filter model in discrete time.

Analogous to the discrete-time case, the continuous-time random process $x(t)$ and the observation $z(t)$ are given by

$$\dot{x}(t) = F(t)x(t) + G(t)w(t), \tag{5.106}$$

$$z(t) = H(t)x(t) + v(t), \tag{5.107}$$

$$\mathrm{E}w(t) = \mathrm{E}v(t) = 0,$$

$$\mathrm{E}w(t_1)w^{\mathrm{T}}(t_2) = Q(t)\delta(t_2 - t_1), \tag{5.108}$$

$$\mathrm{E}v(t_1)v^{\mathrm{T}}(t_2) = R(t)\delta(t_2 - t_1), \tag{5.109}$$

$$\mathrm{E}w(t)v^{\mathrm{T}}(\eta) = 0, \tag{5.110}$$

where $F(t)$, $G(t)$, $H(t)$, $Q(t)$, and $R(t)$ are $n \times n$, $n \times n$, $\ell \times n$, $n \times n$, and $\ell \times \ell$ 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 $z(t)$, $0 \leq t \leq T$ that minimizes the scalar equation

$$\mathrm{E}[x(t) - \hat{x}(t)]^{\mathrm{T}} M[x(t) - \hat{x}(t)], \tag{5.111}$$

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 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 4, 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 5.24 and 5.26 can be combined. By substituting the above relations, one can get the result

$$\begin{aligned} P_{k(-)} &= [I + F(t)\Delta t][I - \overline{K}_{k-1}H_{k-1}]P_{k-1(-)} \\ &\quad \times [I + F(t)\Delta t]^{\mathrm{T}} + G(t)Q(t)G^{\mathrm{T}}(t)\Delta t, \end{aligned} \tag{5.112}$$

$$\begin{aligned} \frac{P_{k(-)} - P_{k-1(-)}}{\Delta t} &= F(t)P_{k-1(-)} + P_{k-1(-)}F^{\mathrm{T}}(t) \\ &\quad + G(t)Q(t)G^{\mathrm{T}}(t) - \frac{\overline{K}_{k-1}H_{k-1}P_{k-1(-)}}{\Delta t} \\ &\quad - F(t)\overline{K}_{k-1}H_{k-1}P_{k-1(-)}F^{\mathrm{T}}(t)\Delta t \\ &\quad + \text{higher order terms}. \end{aligned} \tag{5.113}$$

The Kalman gain of Equation 5.19 becomes, in the limit,

$$\lim_{\Delta t\to 0}\left[\frac{\overline{K}_{k-1}}{\Delta t}\right] = \lim_{\Delta t\to 0}\{P_{k-1(-)}H_{k-1}^{\mathrm{T}}[H_{k-1}P_{k-1(-)}H_{k-1}^{\mathrm{T}}\Delta t + R(t)]^{-1}\}$$
$$= PH^{\mathrm{T}}R^{-1} = \overline{K}(t). \tag{5.114}$$

Substituting Equation 5.114 in 5.113 and taking the limit as $\Delta t \to 0$, one obtains the desired result

$$\dot{P}(t) = F(t)P(t) + P(t)F^{\mathrm{T}}(t) + G(t)Q(t)G^{\mathrm{T}}(t)$$
$$- P(t)H^{\mathrm{T}}(t)R^{-1}(t)H(t)P(t) \tag{5.115}$$

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 5.8. 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 5.115 to the form

$$\dot{P}(t) = F(t)P(t) + P(t)F^{\mathrm{T}}(t) + G(t)Q(t)G^{\mathrm{T}}(t) - \overline{K}(t)R(t)\overline{K}^{\mathrm{T}}(t). \tag{5.116}$$

In a similar manner, the state vector update equation can be derived from Equations 5.21 and 5.25 by taking the limit as $\Delta t \to 0$ to obtain the differential equation for the estimate:

$$\hat{x}_{k(+)} = \Phi_{k-1}\hat{x}_{k-1(+)} + \overline{K}[z_k - H_k\Phi_{k-1}x_{k-1(+)}] \tag{5.117}$$
$$\approx [I + F\ \Delta t]\ \hat{x}_{k-1(+)} + \overline{K}_k[z_k - H_k(I + F\ \Delta t)\hat{x}_{k-1(+)}] \tag{5.118}$$
$$\dot{\hat{x}}(t_k) = \lim_{\Delta t\to 0}\frac{x_{k(+)} - x_{k-1(+)}}{\Delta t} \tag{5.119}$$
$$= \lim_{\Delta t\to 0}\left[F\hat{x}_{k-1(+)}\frac{\overline{K}_k}{\Delta t}\left(z_k - H_k\hat{x}_{k-1(+)} - H_kF_k\ \Delta t\ \hat{x}_{k-1(+)}\right)\right] \tag{5.120}$$
$$\dot{\hat{x}}(t_k) = \lim_{\Delta t\to 0}\frac{x_{k(+)} - x_{k-1(+)}}{\Delta t} \tag{5.121}$$
$$= \lim_{\Delta t\to 0}\left[F\hat{x}_{k-1(+)}\frac{\overline{K}_k}{\Delta t}\left(z_k - H_k\hat{x}_{k-1(+)} - H_kF_k\ \Delta t\ \hat{x}_{k-1(+)}\right)\right] \tag{5.122}$$
$$= F(t)\hat{x}(t) + \overline{K}(t)[z(t) - H(t)\hat{x}(t)] \tag{5.123}$$

with initial condition $\hat{x}(0)$. Equations 5.114, 5.116, and 5.123 define the continuous-time Kalman estimator, which is also called the *Kalman–Bucy filter* [28, 33–35].

5.4 OPTIMAL LINEAR PREDICTORS

5.4.1 Prediction as Filtering

Prediction is equivalent to filtering when the measurement data are not available or are unreliable. In such cases, the Kalman gain matrix $\overline{K}_k$ is forced to be zero. Hence, Equations 5.21, 5.25, and 5.123 become

$$\hat{x}_{k(+)} = \Phi_{k-1}\hat{x}_{(k-1)(+)} \tag{5.124}$$

and

$$\dot{\hat{x}}(t) = F(t)\hat{x}(t). \tag{5.125}$$

Previous values of the estimates will become the initial conditions for the above equations.

5.4.2 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 the filter can continue to operate without modification. One can continue using the prediction algorithm given in Section 5.4 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.

5.5 CORRELATED NOISE SOURCES

5.5.1 Correlation between Plant and Measurement Noise

We want to consider the extensions of the results given in Sections 5.2 and 5.3, allowing correlation between the two noise processes. Let the correlation be given by

$$\mathrm{E}\langle w_{k_1} v_{k_2}^{\mathrm{T}} \rangle = C_k \Delta(k_2 - k_1), \quad \text{for the discrete-time case,}$$

$$\mathrm{E}\langle w(t_1) v^{\mathrm{T}}(t_2) \rangle = C(t)\delta(t_2 - t_1), \quad \text{for the continuous-time case.}$$

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 5.3 have been modified as

$$\overline{K}_k = [P_{k(-)}H_k^{\mathrm{T}} + C_k][H_k P_{k(-)} H_k^{\mathrm{T}} + R_k + H_k C_k + C_k^{\mathrm{T}} H^{\mathrm{T}}]^{-1},$$

$$P_{k(+)} = P_{k(-)} - \overline{K}_k[H_k P_{k(-)} + C_k^{\mathrm{T}}],$$
$$\hat{x}_{k(+)} = \hat{x}_{k(-)} + \overline{K}_k[z_k - H_k \hat{x}_{k(-)}].$$

Similarly, the continuous-time estimator algorithms can be extended to include the correlation. Equation 5.114 is changed as follows [36, 37]:

$$\overline{K}(t) = [P(t)H^{\mathrm{T}}(t) + C(t)]R^{-1}(t).$$

5.5.2 Time-Correlated Measurements

Correlated measurement noise v_k can be modeled by a shaping filter driven by white noise (see Section 5.5). Let the measurement model be given by

$$z_k = H_k x_k + v_k,$$

where

$$v_k = A_{k-1} v_{k-1} + \eta_{k-1} \tag{5.126}$$

and η_k is zero-mean white noise.

Equation 5.1 is augmented by Equation 5.126, and the new state vector $X_k = [x_k \;\; v_k]^{\mathrm{T}}$ satisfies the difference equation:

$$X_k = \begin{bmatrix} x_k \\ -- \\ v_k \end{bmatrix} = \begin{bmatrix} \Phi_{k-1} & 0 \\ --- & --- \\ 0 & A_{k-1} \end{bmatrix} \begin{bmatrix} x_{k-1} \\ -- \\ v_{k-1} \end{bmatrix} + \begin{bmatrix} w_{k-1} \\ --- \\ \eta_{k-1} \end{bmatrix},$$
$$z_k = [H_k \vdots I]X_k.$$

The measurement noise is zero, $R_k = 0$. The estimator algorithm will work as long as $H_k P_{k(-)} H_k^{\mathrm{T}} + R_k$ is invertible. Details of numerical difficulties of this problem (when R_k is singular) are given in Chapter 7.

For continuous-time estimators, the augmentation does not work because $\overline{K}(t) = P(t)H^{\mathrm{T}}(t)R^{-1}(t)$ is required. Therefore, $R^{-1}(t)$ must exist. Alternate techniques are required. For detailed information, see Gelb et al. [9].

5.6 RELATIONSHIPS BETWEEN KALMAN 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 5.3, 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 $t \to \infty$, then $\dot{P}(t) \to 0$. The Riccati differential equation from Section 5.3 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 [12].

5.7 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 $\hat{x} = \mathrm{E}\langle x\rangle$. That is, the estimated value is the *mean* of the probability distribution of the state.

5.7.1 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.

[6]These are concepts from decision theory, which includes estimation theory. The theory might have been built 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.

5.7.1.1 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.

5.7.1.2 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

$$L(\hat{x} - x) = (\hat{x} - x)^{\mathrm{T}} M (\hat{x} - x), \tag{5.127}$$

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 assure that the loss is zero only if the error is zero, and loss is a monotonically increasing function of the absolute estimation error.

5.7.2 Expected Value of a Quadratic Loss Function

5.7.2.1 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 $\hat{x} - \mathrm{E}\langle x \rangle$, where $\mathrm{E}\langle \hat{x} \rangle = \mathrm{E}\langle x \rangle$. This demonstration will depend upon the following identities:

$$\hat{x} - x = (\hat{x} - \mathrm{E}\langle x \rangle) - (x - \mathrm{E}\langle x \rangle), \tag{5.128}$$

$$\mathop{\mathrm{E}}_{x}\langle x - \mathrm{E}\langle x \rangle \rangle = 0, \tag{5.129}$$

$$\begin{aligned} \mathop{\mathrm{E}}_{x}\langle (x - \mathrm{E}\langle x \rangle)^{\mathrm{T}} M (x - \mathrm{E}\langle x \rangle) \rangle & \\ = \mathop{\mathrm{E}}_{x}\langle \mathrm{trace}[(x - \mathrm{E}\langle x \rangle)^{\mathrm{T}} M (x - \mathrm{E}\langle x \rangle)] \rangle & \quad (5.130) \\ = \mathop{\mathrm{E}}_{x}\langle \mathrm{trace}[M (x - \mathrm{E}\langle x \rangle)(x - \mathrm{E}\langle x \rangle)^{\mathrm{T}}] \rangle & \quad (5.131) \\ = \mathrm{trace}[M \mathop{\mathrm{E}}_{x}\langle (x - \mathrm{E}\langle x \rangle)(x - \mathrm{E}\langle x \rangle)^{\mathrm{T}}] & \quad (5.132) \\ = \mathrm{trace}[MP], & \quad (5.133) \end{aligned}$$

$$P \overset{\mathrm{def}}{=} \mathop{\mathrm{E}}_{x}\langle (x - \mathrm{E}\langle x \rangle)(x - \mathrm{E}\langle x \rangle)^{\mathrm{T}} \rangle. \tag{5.134}$$

5.7.2.2 Risk of a Quadratic Loss Function In the case of the quadratic loss function defined above, the expected loss (risk) will be

$$\begin{aligned} \mathcal{R}(\hat{x}) &= \mathop{\mathrm{E}}_{x}\langle L(\hat{x} - x) \rangle & (5.135) \\ &= \mathop{\mathrm{E}}_{x}\langle (\hat{x} - x)^{\mathrm{T}} M (\hat{x} - x) \rangle & (5.136) \end{aligned}$$

$$= \underset{x}{\mathrm{E}}\langle[(\hat{x} - \mathrm{E}\langle x\rangle) - (x - \mathrm{E}\langle x\rangle)]^{\mathrm{T}} M[(\hat{x} - \mathrm{E}\langle x\rangle) - (x - \mathrm{E}\langle x\rangle)]\rangle \tag{5.137}$$

$$= \underset{x}{\mathrm{E}}\langle(\hat{x} - \mathrm{E}\langle x\rangle)^{\mathrm{T}} M(\hat{x} - \mathrm{E}\langle x\rangle) + (x - \mathrm{E}\langle x\rangle)^{\mathrm{T}} M(x - \mathrm{E}\langle x\rangle)\rangle$$
$$- \underset{x}{\mathrm{E}}\langle(\hat{x} - \mathrm{E}\langle x\rangle)^{\mathrm{T}} M(x - \mathrm{E}\langle x\rangle) + (x - \mathrm{E}\langle x\rangle)^{\mathrm{T}} M(\hat{x} - \mathrm{E}\langle x\rangle)\rangle \tag{5.138}$$

$$= (\hat{x} - \mathrm{E}\langle x\rangle)^{\mathrm{T}} M(\hat{x} - \mathrm{E}\langle x\rangle) + \underset{x}{\mathrm{E}}\langle(x - \mathrm{E}\langle x\rangle)^{\mathrm{T}} M(x - \mathrm{E}\langle x\rangle)\rangle$$
$$- (\hat{x} - \mathrm{E}\langle x\rangle)^{\mathrm{T}} M_{x}\langle(x - \mathrm{E}\langle x\rangle)\rangle - \underset{x}{\mathrm{E}}\langle(x - \mathrm{E}\langle x\rangle)\rangle^{\mathrm{T}} M(\hat{x} - \mathrm{E}\langle x\rangle) \tag{5.139}$$

$$= (\hat{x} - \mathrm{E}\langle x\rangle)^{\mathrm{T}} M(\hat{x} - \mathrm{E}\langle x\rangle) + \mathrm{trace}[MP], \tag{5.140}$$

which is a quadratic function of $\hat{x} - \mathrm{E}\langle x\rangle$ with the added nonnegative[7] constant trace$[MP]$.

5.7.3 Unbiased Estimates and Quadratic Loss

The *estimate* $\hat{x} = \mathrm{E}\langle x\rangle$ minimizes the expected value of any *positive-definite quadratic loss function*. From the above derivation,

$$\mathcal{R}(\hat{x}) \geq \mathrm{trace}[MP] \tag{5.141}$$

and

$$\mathcal{R}(\hat{x}) = \mathrm{trace}[MP] \tag{5.142}$$

only if

$$\hat{x} = \mathrm{E}\langle x\rangle, \tag{5.143}$$

where it has been assumed only that the mean $\mathrm{E}\langle x\rangle$ and covariance $\underset{x}{\mathrm{E}}\langle(x - \mathrm{E}\langle x\rangle)(x - \mathrm{E}\langle x\rangle)^{\mathrm{T}}\rangle$ 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).

5.7.3.1 Unbiased Estimates An estimate $\hat{x}$ is called *unbiased* if the expected estimation error $\underset{x}{\mathrm{E}}\langle\hat{x} - x\rangle = 0$. What has just been shown is that an unbiased estimate minimizes the expected value of any quadratic loss function of estimation error.

5.8 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. [20].

[7]Recall that M and P are symmetric and nonnegative definite, and the matrix trace of any product of symmetric nonnegative definite matrices is nonnegative.

5.8.1 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.

5.8.1.1 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.

5.8.1.2 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.

5.8.1.3 Derivation By taking the derivative of the matrix fraction $A(t)B^{-1}(t)$ with respect to t and using the fact[8] that

$$\frac{\mathrm{d}}{\mathrm{d}t}B^{-1}(t) = -B^{-1}(t)\dot{B}(t)B^{-1}(t),$$

one can arrive at the following decomposition of the matrix Riccati differential equation, where GQG^{T} has been reduced to an equivalent Q:

$$\begin{aligned}
&\dot{A}(t)B^{-1}(t) - A(t)B^{-1}(t)\dot{B}(t)B^{-1}(t) \\
&\qquad = \frac{d}{dt}\{A(t)B^{-1}(t)\} && (5.144)\\
&\qquad = \frac{d}{dt}P(t) && (5.145)\\
&\qquad = F(t)P(t) + P(t)F^{\mathrm{T}}(t) \\
&\qquad\quad - P(t)H^{\mathrm{T}}(t)R^{-1}(t)H(t)P(t) + Q(t) && (5.146)
\end{aligned}$$

[8]This formula is derived in Appendix B (in the Wiley web site).

$$= F(t)A(t)B^{-1}(t) + A(t)B^{-1}(t)F^{\mathrm{T}}(t) - A(t)B^{-1}(t)H^{\mathrm{T}}(t)R^{-1}(t)H(t)A(t)B^{-1}(t) + Q(t), \tag{5.147}$$

$$\dot{A}(t) - A(t)B^{-1}(t)\dot{B}(t) = F(t)A(t) + A(t)B^{-1}(t)F^{\mathrm{T}}(t)B(t) - A(t)B^{-1}(t)H^{\mathrm{T}}(t)R^{-1}(t)H(t)A(t) + Q(t)B(t), \tag{5.148}$$

$$\dot{A}(t) - A(t)B^{-1}(t)\{\dot{B}(t)\} = F(t)A(t) + Q(t)B(t) - A(t)B^{-1}(t) \times \{H^{\mathrm{T}}(t)R^{-1}(t)H(t)A(t) - F^{\mathrm{T}}(t)B(t)\}, \tag{5.149}$$

$$\dot{A}(t) = F(t)A(t) + Q(t)B(t), \tag{5.150}$$

$$\dot{B}(t) = H^{\mathrm{T}}(t)R^{-1}(t)H(t)A(t) - F^{\mathrm{T}}(t)B(t), \tag{5.151}$$

$$\frac{d}{dt}\begin{bmatrix} A(t) \\ B(t) \end{bmatrix} = \begin{bmatrix} F(t) & Q(t) \\ H^{\mathrm{T}}(t)R^{-1}(t)H(t) & -F^{\mathrm{T}}(t) \end{bmatrix} \begin{bmatrix} A(t) \\ B(t) \end{bmatrix}. \tag{5.152}$$

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.

5.8.1.4 Hamiltonian Matrix This is the name[9] given to the matrix

$$\Psi(t) = \begin{bmatrix} F(t) & Q(t) \\ H^{\mathrm{T}}(t)R^{-1}(t)H(t) & -F^{\mathrm{T}}(t) \end{bmatrix} \tag{5.153}$$

of the matrix Riccati differential equation.

5.8.1.5 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.

5.8.2 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.

[9] After the Irish mathematician and physicist William Rowan Hamilton (1805–1865).

5.8.3 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*.

5.8.3.1 Linearizing the Differential Equation The scalar time-invariant Riccati differential equation and its linearized equivalent are

$$\dot{P}(t) = FP(t) + P(t)F^{\mathrm{T}} - P(t)HR^{-1}HP(t) + Q,$$
$$\begin{bmatrix} \dot{A}(t) \\ \dot{B}(t) \end{bmatrix} = \begin{bmatrix} F & Q \\ H^{\mathrm{T}}R^{-1}H & -F^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} A(t) \\ B(t) \end{bmatrix},$$

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*.

5.8.3.2 Fundamental Solution of Linear Time-Invariant Differential Equation The linear time-invariant differential equation has the general solution

$$\begin{bmatrix} A(t) \\ B(t) \end{bmatrix} = e^{\Psi t} \begin{bmatrix} P(0) \\ 1 \end{bmatrix}, \tag{5.154}$$
$$\Psi = \begin{bmatrix} F & Q \\ \dfrac{H^2}{R} & -F \end{bmatrix}.$$

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

$$M^{-1} = \begin{bmatrix} \dfrac{-H^2}{2\phi R} & \dfrac{H^2 Q}{2H^2Q + 2F^2R - 2F\phi R} \\ \dfrac{H^2}{2\phi R} & \dfrac{H^2 Q}{2H^2Q + 2F^2R + 2F\phi R} \end{bmatrix},$$

by which it can be diagonalized as

$$M^{-1}\Psi M = \begin{bmatrix} \lambda^2 & 0 \\ 0 & \lambda_1 \end{bmatrix},$$
$$\lambda_2 = -\frac{H^2Q + F^2R}{\phi R}, \quad \lambda_1 = \frac{H^2Q + F^2R}{\phi R},$$

with the characteristic values of Ψ along its diagonal. The exponential of the diagonalized matrix, multiplied by t, will be

$$e^{M^{-1}\Psi Mt} = \begin{bmatrix} e^{\lambda_2 t} & 0 \\ 0 & e^{\lambda_1 t} \end{bmatrix}.$$

Using this, one can write the fundamental solution of the linear homogeneous time-invariant equation as

$$\begin{aligned} e^{\Psi t} &= \sum_{k=0}^{\infty} \frac{1}{k!} t^k \Psi^k \\ &= M\left(\sum_{k=0}^{\infty} \frac{1}{k!}[M^{-1}\Psi M]^k\right) M^{-1} \\ &= Me^{M^{-1}\Psi Mt}M^{-1} \\ &= M\begin{bmatrix} e^{\lambda_2 t} & 0 \\ 0 & e^{\lambda_1 t} \end{bmatrix} M^{-1} \\ &= \frac{1}{2e^{\phi t}\phi}\begin{bmatrix} \phi(\psi(t)+1) + F(\psi(t)-1) & Q(1-\psi(t)) \\ \dfrac{H^2(\psi(t)-1)}{R} & F(1-\psi(t)) + \phi(1+\psi(t)) \end{bmatrix}, \\ \psi(t) &= e^{2\phi t} \end{aligned}$$

and the solution of the linearized system as

$$\begin{aligned} \begin{bmatrix} A(t) \\ B(t) \end{bmatrix} &= e^{\Psi t}\begin{bmatrix} P(0) \\ 1 \end{bmatrix} \\ &= \frac{1}{2e^{\phi t}\phi}\begin{bmatrix} P(0)[\phi(\psi(t)+1) + F(\psi(t)-1)] - \dfrac{Q(\psi(t)-1)}{R^2} \\ \dfrac{P(0)H^2(\psi(t)-1)}{R} - \phi(\psi(t)+1) - F(\psi(t)-1) \end{bmatrix}. \end{aligned}$$

5.8.3.3 General Solution of Scalar Time-Invariant Riccati Equation The general solution formula may now be composed from the previous results as

$$\begin{aligned} P(t) &= A(t)/B(t) \\ &= \frac{\mathcal{N}_P(t)}{\mathcal{D}_P(t)}, \end{aligned} \tag{5.155}$$

$$\begin{aligned} \mathcal{N}_P(t) &= R[P(0)(\phi + F) + Q] + R[P(0)(\phi - F) - Q]e^{-2\phi t} \\ &= R\left[P(0)\left(\sqrt{F^2 + \frac{H^2Q}{R}} + F\right) + Q\right] \\ &\quad + R\left[P(0)\left(\sqrt{F^2 + \frac{H^2Q}{R}} - F\right) - Q\right]e^{-2\phi t}, \end{aligned} \tag{5.156}$$

$$\begin{aligned} \mathcal{D}_P(t) &= [H^2P(0) + R(\phi - F)] - [H^2P(0) - R(F + \phi)]e^{-2\phi t} \\ &= \left[H^2P(0) + R\left(\sqrt{F^2 + \frac{H^2Q}{R}} - F\right)\right] \\ &\quad - \left[H^2P(0) - R\left(\sqrt{F^2 + \frac{H^2Q}{R}} + F\right)\right]e^{-2\phi t}. \end{aligned} \tag{5.157}$$

5.8.3.4 Singular Values of 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.)

5.8.3.5 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:

$$\lim_{t\to 0} N_P(t) = 2P(0)R\sqrt{F^2 + \frac{H^2Q}{R}},$$

$$\lim_{t\to 0} D_P(t) = 2R\sqrt{F^2 + \frac{H^2Q}{R}},$$

$$\lim_{t\to 0} P(t) = P(0),$$

$$\lim_{t\to\infty} P(t) = \frac{R}{H^2}\left(F + \sqrt{F^2 + \frac{H^2Q}{R}}\right). \tag{5.158}$$

5.8.4 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.

5.8.4.1 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

$$\tau(F, H, R, Q) = 2\sqrt{F^2 + \frac{H^2Q}{R}} \tag{5.159}$$

of the problem parameters. Note that this function does not depend upon the initial value of P.

5.8.4.2 Asymptotic and Steady-State Solutions The asymptotic solution of the scalar time-invariant Riccati differential equation as $t \to \infty$ is given in Equation 5.158. 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 5.158 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.

[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.

5.8.4.3 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.

5.8.4.4 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 5.7, which is a multiplot of solutions to 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

$$P(t) = \frac{e^{2t}[1 + P(0)] - [1 - P(0)]}{e^{2t}[1 + P(0)] + [1 - P(0)]}$$

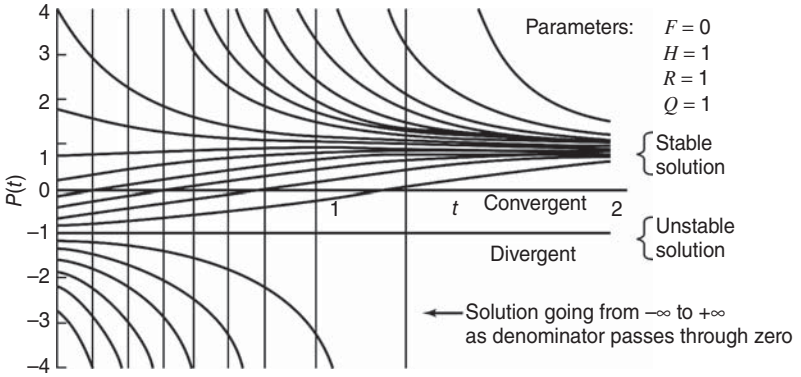


Figure 5.7 Solutions of the scalar time-invariant Riccati equation.

in terms of the initial value $P(0)$. Solutions of the initial-value problem with different initial values are plotted over the time interval $0 \le t \le 2$. 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.

5.8.4.5 Convergent and Divergent Regions The line at $P = -1$ in Figure 5.7 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.

5.8.5 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 $(F^2 + H^2Q/R)^{1/2}$, 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

$$P(\infty) = \frac{R}{H^2}\left(F + \sqrt{F^2 + \frac{H^2Q}{R}}\right),$$

which is influenced by both the sign *and* 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 [38].

5.8.5.1 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.

5.8.6 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 5.6 (Solving the Algebraic Riccati Equation in Continuous Time for the Harmonic Resonator Problem) The problem is to characterize the asymptotic uncertainty in estimating the state (position and velocity) of a damped harmonic resonator driven by white 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

$$\begin{aligned}
0 &= FP + PF^{\mathrm{T}} - PH^{\mathrm{T}}R^{-1}HP + Q,\\
F &= \begin{bmatrix} 0 & 1 \\ -\dfrac{1+\omega^2\tau^2}{\tau^2} & \dfrac{-2}{\tau} \end{bmatrix},\\
H &= \begin{bmatrix} 1 & 0 \end{bmatrix},\\
Q &= \begin{bmatrix} 0 & 0 \\ 0 & q \end{bmatrix},
\end{aligned}$$

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}

$$p_{12} = \frac{p_{11}^2}{2R},$$
$$p_{22} = \frac{Rq\tau^2 - \tau^2 p_{12}^2 - 2R(1+\omega^2\tau^2)p_{12}}{4R\tau}$$

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 nonnegative covariance matrix P:

$$p_{11} = \frac{R(1-b)}{\tau},$$
$$p_{12} = \frac{R(1-b)^2}{2\tau^2},$$
$$p_{22} = \frac{R}{\tau^3}(-6+2 \quad \omega^2\tau^2 - 4a + (4+a)b),$$
$$a = \sqrt{(1+\omega^2\tau^2)^2 + \frac{q\tau^4}{R}}, \quad b = \sqrt{2(1-\omega^2\tau^2+a)}.$$

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 us 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.

5.8.7 Newton–Raphson Solution of the Algebraic Riccati Differential Equation

The *Newton–Raphson solution* of n differentiable functional equations

$$0 = f_1(x_1, x_2, x_3, \ldots, x_n),$$
$$0 = f_2(x_1, x_2, x_3, \ldots, x_n),$$
$$0 = f_3(x_1, x_2, x_3, \ldots, x_n),$$

$$\vdots$$

$$0 = f_n(x_1, x_2, x_3, \ldots, x_n)$$

in n unknowns $x_1, x_2, x_3, \ldots, x_n$ is the iterative vector procedure

$$x \leftarrow x - \mathcal{F}^{-1} f(x) \tag{5.160}$$

using the vector and matrix variables

$$x = \begin{bmatrix} x_1 & x_2 & x_3 & \cdots & x_n \end{bmatrix}^{\mathrm{T}},$$

$$f(x) = \begin{bmatrix} f_1(x) & f_2(x) & f_3(x) & \cdots & f_n(x) \end{bmatrix}^{\mathrm{T}},$$

$$\mathcal{F} = \begin{bmatrix} \dfrac{\partial f_1}{\partial x_1} & \dfrac{\partial f_1}{\partial x_2} & \dfrac{\partial f_1}{\partial x_3} & \cdots & \dfrac{\partial f_1}{\partial x_n} \\ \dfrac{\partial f_2}{\partial x_1} & \dfrac{\partial f_2}{\partial x_2} & \dfrac{\partial f_2}{\partial d_3} & \cdots & \dfrac{\partial f_2}{\partial x_n} \\ \dfrac{\partial f_3}{\partial x_1} & \dfrac{\partial f_3}{\partial x_2} & \dfrac{\partial f_3}{\partial x_3} & \cdots & \dfrac{\partial f_3}{\partial x_n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \dfrac{\partial f_n}{\partial x_1} & \dfrac{\partial f_n}{\partial x_2} & \dfrac{\partial f_n}{\partial x_3} & \cdots & \dfrac{\partial f_n}{\partial x_n} \end{bmatrix}.$$

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 the other 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 and then show the form that the variables of the Newton–Raphson solution will take for this vectorization.

5.8.7.1 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

$$1 \le j \le n,$$

$$1 \le i \le j,$$
$$p = \frac{1}{2}j(j-1) + i,$$
$$1 \le p \le \frac{1}{2}n(n+1).$$

Similarly, the upper-triangular part of P can be mapped into a singly subscripted array x with index q, according to the rules

$$1 \le \ell \le n,$$
$$1 \le k \le \ell,$$
$$q = \frac{1}{2}\ell(\ell-1) + k,$$
$$1 \le q \le \frac{1}{2}n(n+1),$$

whereby $P_{k\ell}$ is mapped into x_q.

5.8.7.2 Values of Variables for Newton–Raphson Solution of Steady-State Matrix Riccati Differential Equation The solution is an implementation of the recursion formula 5.74 with

$$f_p = Z_{ij}, \tag{5.161}$$
$$Z = FP + PF^{\mathrm{T}} - PH^{\mathrm{T}}R^{-1}HP + Q, \tag{5.162}$$
$$x_q = P_{k\ell}, \tag{5.163}$$
$$p = \frac{1}{2}j(j-1) + i, \tag{5.164}$$
$$q = \frac{1}{2}\ell(\ell-1) + k, \tag{5.165}$$
$$\begin{aligned}\mathcal{F}_{pq} &= \frac{\partial f_p}{\partial x_q} \\ &= \frac{\partial Z_{ij}}{\partial P_{k\ell}} \\ &= \Delta_{j\ell}\mathcal{S}_{ik} + \Delta_{ik}\mathcal{S}_{j\ell},\end{aligned} \tag{5.166}$$
$$\mathcal{S} = F - PH^{\mathrm{T}}R^{-1}H, \tag{5.167}$$
$$\Delta_{ab} \overset{\text{def}}{=} \begin{cases} 1 & \text{if} \quad a = b \\ 0 & \text{if} \quad a \ne b. \end{cases} \tag{5.168}$$

The least obvious of these is Equation 5.166, which will now be derived.

5.8.7.3 "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 zeros elsewhere. As a vector, it has the property that

$$MI_{\cdot j} = M_{\cdot j}$$

for any conformable matrix M.

5.8.7.4 Matrix Partial Derivatives With this notation, one can write matrix partial derivatives as follows:

$$\frac{\partial P}{\partial P_{k\ell}} = I_{\cdot k} I_{\cdot \ell}^{\mathrm{T}}, \tag{5.169}$$

$$\frac{\partial Z}{\partial P_{k\ell}} = F\frac{\partial P}{\partial P_{k\ell}} + \frac{\partial P}{\partial P_{k\ell}}F^{\mathrm{T}} - \frac{\partial P}{\partial P_{k\ell}}H^{\mathrm{T}}R^{-1}HP - PH^{\mathrm{T}}R^{-1}H\frac{\partial P}{\partial P_{k\ell}} \tag{5.170}$$

$$= FI_{\cdot k}I_{\cdot \ell}^{\mathrm{T}} + I_{\cdot k}I_{\cdot \ell}^{\mathrm{T}}F^{\mathrm{T}} - I \cdot kI_{\cdot \ell}^{\mathrm{T}}H^{\mathrm{T}}R^{-1}HP - PH^{\mathrm{T}}R^{-1}H\frac{\partial \partial}{\partial P_{k\ell}} \tag{5.171}$$

$$= F_{\cdot k}I_{\cdot \ell}^{\mathrm{T}} + I_{\cdot k}F_{\cdot \ell}^{\mathrm{T}} - I_{\cdot k}M_{\cdot \ell}^{\mathrm{T}} - M_{\cdot k}I_{\cdot \ell}^{\mathrm{T}} \tag{5.172}$$

$$= (F - M)_{\cdot k}I_{\cdot \ell}^{\mathrm{T}} + I_{\cdot k}(F - M)_{\cdot \ell}^{\mathrm{T}} \tag{5.173}$$

$$= S_{\cdot k}I_{\cdot \ell}^{\mathrm{T}} + I_{\cdot k}S_{\cdot \ell}^{\mathrm{T}}, \tag{5.174}$$

$$S = F - M, \tag{5.175}$$

$$M = PH^{\mathrm{T}}R^{-1}H. \tag{5.176}$$

Note that on the right-hand side of Equation 5.174, 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 5.166. This completes its derivation.

5.8.7.5 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.

5.8.8 MacFarlane–Potter–Fath Eigenstructure Method

5.8.8.1 Steady-State Solution of Time-Invariant Matrix Riccati Differential Equation It was discovered independently by MacFarlane [39], Potter [40], and Fath [41] that the solution $P(\infty)$ of the continuous-time form of the steady-state matrix Riccati differential equation can be expressed in the form

$$P(\infty) = AB^{-1},$$

$$\begin{bmatrix} A \\ B \end{bmatrix} = \begin{bmatrix} e_{i_1} & e_{i_2} & e_{i_3} & \cdots & e_{i_n} \end{bmatrix},$$

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 5.1 If A and B are $n \times n$ matrices such that B is nonsingular and

$$\Psi_c \begin{bmatrix} A \\ B \end{bmatrix} = \begin{bmatrix} A \\ B \end{bmatrix} D \tag{5.177}$$

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 5.177 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 that 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.)

5.9 MATRIX RICCATI EQUATION IN DISCRETE TIME

5.9.1 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.

5.9.2 Matrix Fraction Propagation of the *a priori* Covariance

Lemma 5.2 If the state-transition matrices Φ_k are nonsingular and

$$P_{k(-)} = A_k B_k^{-1} \tag{5.178}$$

is a nonsingular matrix solution of the discrete-time Riccati equation at time t_k, then

$$P_{k+1}(-) = A_{k+1} B_{k+1}^{-1} \tag{5.179}$$

is a solution at time t_{k+1}, where

$$\begin{bmatrix} A_{k+1} \\ B_{k+1} \end{bmatrix} = \begin{bmatrix} Q_k & I \\ I & 0 \end{bmatrix} \begin{bmatrix} \Phi_k^{-\mathrm{T}} & 0 \\ 0 & \Phi_k \end{bmatrix} \begin{bmatrix} H_k^{\mathrm{T}} R_k^{-1} H_k & I \\ I & 0 \end{bmatrix} \begin{bmatrix} A_k \\ B_k \end{bmatrix} \tag{5.180}$$

$$= \begin{bmatrix} \Phi_k + Q_k \Phi_k^{-\mathrm{T}} H_k^{\mathrm{T}} R_k^{-1} H_k & Q_k \Phi_k^{-\mathrm{T}} \\ \Phi_k^{-\mathrm{T}} H_k^{\mathrm{T}} R_k^{-1} H_k & \Phi_k^{-\mathrm{T}} \end{bmatrix} \begin{bmatrix} A_k \\ B_k \end{bmatrix}. \tag{5.181}$$

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}:

$$
\begin{aligned}
A_{k+1}B_{k+1}^{-1} &= \{[\Phi_k + Q_k\Phi_k^{-\mathrm{T}}H_k^{-\mathrm{T}}R_k^{-1}H_k]A_k + Q_k\Phi_k^{-\mathrm{T}}B_k\} \\
&\quad \times \{\Phi_k^{-\mathrm{T}}[H_k^{\mathrm{T}}R_k^{-1}H_kA_kB_k^{-1} + I]B_k\}^{-1} \qquad \text{(definition)} \\
&= \{[\Phi_k + Q_k\Phi_k^{-\mathrm{T}}H_k^{\mathrm{T}}R_k^{-1}H_k]A_k + Q_k\Phi_k^{-\mathrm{T}}B_k\} \\
&\quad \times B_k^{-1}\{H_k^{\mathrm{T}}R_k^{-1}H_kA_kB_k^{-1} + I\}^{-1}\Phi_k^{\mathrm{T}} \qquad \text{(factor } B_k\text{)} \\
&= \{[\Phi_k + Q_k\Phi_k^{-\mathrm{T}}H_k^{\mathrm{T}}R_k^{-\mathrm{T}}H_k]A_kB_k^{-1} + Q_k\Phi_k^{-\mathrm{T}}\} \\
&\quad \times \{H_k^{\mathrm{T}}R_k^{-1}H_kA_kB_k^{-1} + I\}^{-1}\Phi_k^{\mathrm{T}} \qquad \text{(distribute } B_k\text{)} \\
&= \{[\Phi_k + Q_k\Phi_k^{-\mathrm{T}}H_k^{\mathrm{T}}R_k^{-1}H_k]P_{k(-)} + Q_k\Phi_k^{-\mathrm{T}}\} \\
&\quad \times \{H_k^{\mathrm{T}}R_k^{-1}H_kP_{k(-)} + I\}^{-1}\Phi_k^{\mathrm{T}} \qquad \text{(definition)} \\
&= \{\Phi_kP_{k(-)} + Q_k\Phi_k^{-\mathrm{T}}[H_k^{\mathrm{T}}R_k^{-1}H_kP_{k(-)} + I]\} \\
&\quad \times \{H_k^{\mathrm{T}}R_k^{-\mathrm{T}}H_kP_{k(-)} + I\}^{-1}\Phi_k^{\mathrm{T}} \qquad \text{(regroup)} \\
&= \Phi_kP_{k(-)}\{H_k^{\mathrm{T}}R_k^{-\mathrm{T}}H_kP_{k(-)} + I\}^{-1}\Phi_k^{\mathrm{T}} + Q_k\Phi_k^{-\mathrm{T}}\Phi_k^{\mathrm{T}} \quad \text{(distribute)} \\
&= \Phi_k\{H_k^{\mathrm{T}}R_k^{-1}H_k + P_k^{-1}\}^{-1}\Phi_k^{\mathrm{T}} + Q_k \\
&= \Phi_k\{P_{k(-)} - P_{k(-)}H_k^{\mathrm{T}}[H_kP(-)H_k^{\mathrm{T}} + R_k]^{-1} \\
&\quad \times H_kP_{k(-)}\}\Phi_k^{\mathrm{T}} + Q_k \qquad \text{(Hemes)} \\
&= P_{k+1(-)} \qquad \text{(Riccati),}
\end{aligned}
$$

where the "Hemes inversion formula" is given in Appendix B on the companion Wiley web site. 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.

5.9.3 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.

5.9.3.1 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

$$M^{-1} = \begin{bmatrix} -\dfrac{H^2}{\sigma_2 \sigma_1} & \dfrac{H^2 Q - R + \Phi^2 R + \sigma_2 \sigma_1}{2\sigma_2 \sigma_1} \\ \dfrac{H^2}{\sigma_2 \sigma_1} & \dfrac{-(H^2 Q) + R - \Phi^2 R + \sigma_2 \sigma_1}{2\sigma_2 \sigma_1} \end{bmatrix}$$
$$= \frac{1}{4QR\sigma_1 \sigma_2} \begin{bmatrix} \tau_1 \tau_2 & 2QR\tau_1 \\ -\tau_1 \tau_2 & -2QR\tau_2 \end{bmatrix},$$
$$\tau_1 = H^2 Q + R(\Phi^2 - 1) + \sigma, \quad \tau_2 = H^2 Q + R(\Phi^2 - 1) - \sigma.$$

5.9.3.2 Closed-Form Solution This will have the form

$$P_k = A_k B_K^{-\mathrm{T}}$$

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.

5.9.4 MacFarlane–Potter–Fath Eigenstructure Method

5.9.4.1 Steady-State Solution of Time-Invariant Discrete-Time Matrix Riccati Equation The method presented in Section 5.8.8 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 a lemma.

Lemma 5.3 If A and B are $n \times n$ matrices such that B is nonsingular and

$$\Psi_d \begin{bmatrix} A \\ B \end{bmatrix} = \begin{bmatrix} A \\ B \end{bmatrix} D \tag{5.182}$$

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 5.2 that $P_{k+1} = \acute{A}\acute{B}^{-1}$, where

$$\begin{bmatrix} \acute{A} \\ \acute{B} \end{bmatrix} = \begin{bmatrix} (\Phi_k + Q_k\Phi_k^{-\mathrm{T}}H_k^{\mathrm{T}}R_k^{-1}H_k) & Q_k\Phi_k^{-\mathrm{T}} \\ \Phi_k^{-\mathrm{T}}H_k^{\mathrm{T}}R_k^{-1}H_k & \Phi_k^{-\mathrm{T}} \end{bmatrix} \begin{bmatrix} A \\ B \end{bmatrix}$$

$$= \Psi_d \begin{bmatrix} A \\ B \end{bmatrix}$$

$$= \begin{bmatrix} A \\ B \end{bmatrix} D$$

$$= \begin{bmatrix} AD \\ BD \end{bmatrix}.$$

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.

5.10 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 8, these formulas will be used to simplify the model equations.

5.10.1 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

$$\acute{x}_k = A_k x_k, \tag{5.183}$$

$$\acute{z}_k = B_k H_k, \tag{5.184}$$

where x and z are the old variables and $\acute{x}$ and $\acute{z}$ are the new state vector and measurement, respectively.

5.10.1.1 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 ℓ of $\acute{z}_k$ is arbitrary and can depend on k.

5.10.2 New Model Equations

With the above assumptions, the corresponding state, measurement, and state uncertainty covariance equations of the Kalman filter model are transformed to the following forms:

$$\acute{x}_{k+1} = \acute{\Phi}_k \acute{x}_k + \acute{\omega}_k, \tag{5.185}$$

$$\acute{z} = \acute{H}_k \acute{x}_k + \acute{\upsilon}_k, \tag{5.186}$$

$$\acute{P}_{k(+)} = \acute{P}_{k(-)} - \acute{P}_{k(-)} \acute{H}_k [\acute{H}_k \acute{P}_{k(-)} \acute{H}_k^{\mathrm{T}} + \acute{R}_k] \acute{H}_k \acute{P}_{k(-)}, \tag{5.187}$$

$$\acute{P}_{k+1(-)} = \acute{\Phi}_k \acute{P}_{k(+)} \acute{\Phi}_k^{\mathrm{T}} + \acute{Q}_k, \tag{5.188}$$

where the new model parameters are

$$\acute{\Phi}_k = A_k \Phi_k A_k^{-1}, \tag{5.189}$$

$$\acute{H}_k = B_k H_k A_k^{-1}, \tag{5.190}$$

$$\acute{Q}_k = \mathrm{E}\langle \acute{\omega}_k \acute{\omega}_k^{\mathrm{T}} \rangle \tag{5.191}$$

$$= A_k Q_k A_k^{\mathrm{T}}, \tag{5.192}$$

$$\acute{R} = \mathrm{E}\langle \acute{\upsilon}_k \acute{\upsilon}_k^{\mathrm{T}} \rangle \tag{5.193}$$

$$= B_k R_k B_k^{\mathrm{T}}, \tag{5.194}$$

and the new state estimation uncertainty covariance matrices are

$$\acute{P}_{k(\pm)} = A_k P_{k(\pm)} A_k^{\mathrm{T}}. \tag{5.195}$$

5.11 SAMPLE APPLICATIONS

The Kalman filter has been applied to inertial navigation [6, 42, 43], sensor calibration [44], radar tracking [8], manufacturing [18], economics [12], signal processing [18], and freeway traffic modeling [45]—to cite a few examples. This section shows some applications of the programs provided on the companion web site. A simple example of a second-order underdamped oscillator is given here to illustrate the application of the equations in Table 5.3. This harmonic oscillator is an approximation of a longitudinal dynamics of an aircraft short period [46].

Example 5.7 (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

$$\ddot{x}_1(t) + 2\zeta w\dot{x}_1(t) + \omega^2 x_1(t) = 12 + w(t)$$

can be written in state-space form via 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 a trajectory of 100 simulated data points with plant noise and measurement noise equal to zero using the following initial condition and parameter values:

$$\begin{bmatrix}x_1(0)\\ x_2(0)\end{bmatrix} = \begin{bmatrix}0 & \text{ft}\\ 0 & \text{ft/s}\end{bmatrix},$$

$$P(0) = \begin{bmatrix}2 & 0\\ 0 & 2\end{bmatrix},$$

$$Q = 4.47(\text{ft/s})^2, \quad R = 0.01(\text{ft})^2,$$

$$\zeta = 0.2, \quad \omega = 5 \quad \text{rad/s}.$$

Equations 5.21, 5.24, 5.25, and 5.26 were programmed in MATLAB® software on a PC (see Appendix A) to estimate $\hat{x}_1(t)$ and $\hat{x}_2(t)$. Figure 5.8 shows the resulting estimates of position and velocity using the noise-free data generated from simulating the above second-order equation. The estimated and actual values are identical in this case. Figure 5.9 shows the corresponding RMS uncertainties in position and velocity (Figure 5.9(a)), correlation coefficient between position and velocity (Figure 5.9(b)), and Kalman gains (Figure 5.9(c)). These results were generated from the accompanying MATLAB program exam57.m described in Appendix A with sample interval equal to 1 s.

Example 5.8 (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 [47]. This example's equations are drawn from IEEE papers [48, 49].

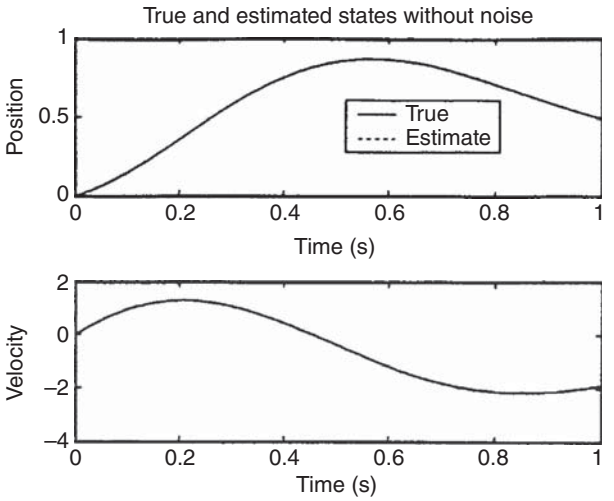


Figure 5.8 Estimated position (ft) and velocity (ft/s) versus time (s).

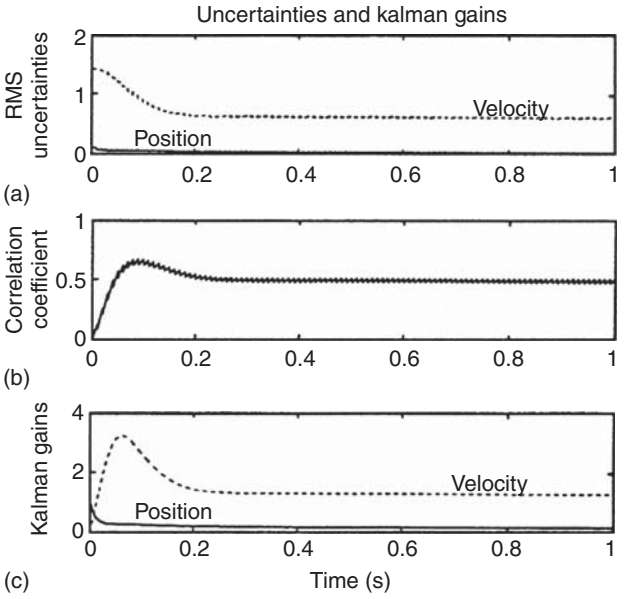


Figure 5.9 RMS uncertainties, position and velocity, correlation coefficient, and Kalman gain.

Difference equations of dynamics 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 is the range of the vehicle at time k

$\dot{r}_k$ is the range rate of the vehicle at time k

U_k^1 is the maneuvering-correlated state noise

θ_k is the bearing of the vehicle at time k

$\dot{\theta}_k$ is the bearing rate of the vehicle at time k

U_k^2 is the maneuvering-correlated state noise

T is the sampling period in seconds

$w_{\mathrm{k}}^{\mathrm{T}} = [w_k^1 w_k^2]$ is the zero-mean white-noise sequences and covariance of σ_1^2 and σ_2^2, respectively

$v_k^{\mathrm{T}} = [v_k^1 v_k^2]$ is the 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:

$$\rho = \text{correlation coefficient} = \frac{\mathrm{E}[U_k U_{k-1}]}{\sigma_m^2} = \begin{cases} 1 - \lambda T, & T \le \dfrac{1}{\lambda}, \\ 0, & T > \dfrac{1}{\lambda}, \end{cases}$$

where σ_m^2 is the maneuver variance and λ 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 5.5 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 $T = 5, 10, 15$ s, respectively. Also,

$$\sigma_r^2 = (1000 \text{ m})^2, \quad \sigma_\theta^2 = (0.017\text{rad})^2,$$
$$\sigma_1^2 = (103/3)^2, \quad \sigma_2^2 = 1.3 \times 10^{-8}.$$

Some parts of this example are discussed in Reference 50. Results of covariances and Kalman gain plots are shown in Figures 5.10–5.12. 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 exam58.m described in Appendix A.

5.12 SUMMARY

5.12.1 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 4. The discrete-time estimator (Kalman filter) has been derived and described, including its implementation equations and

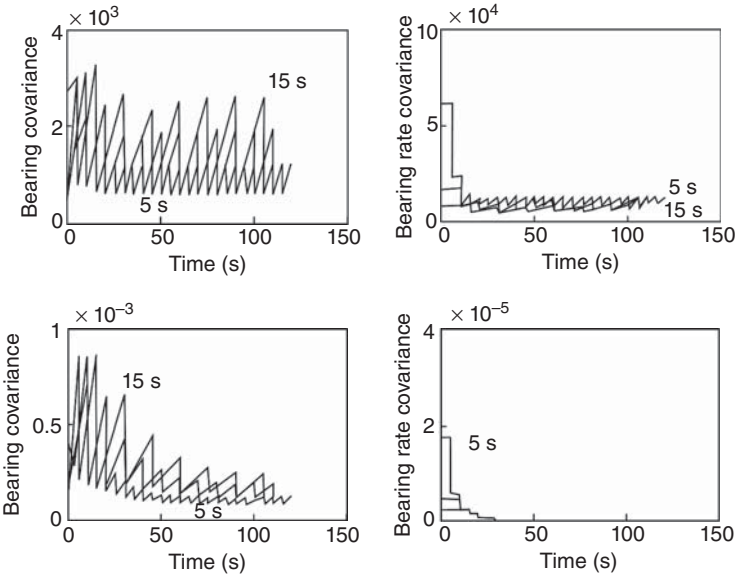


Figure 5.10 Covariances.

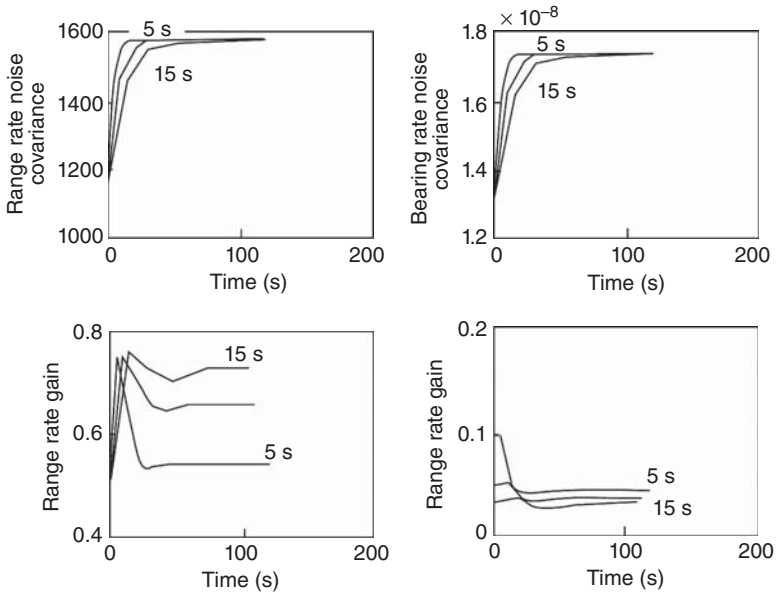


Figure 5.11 Covariances and Kalman gains.

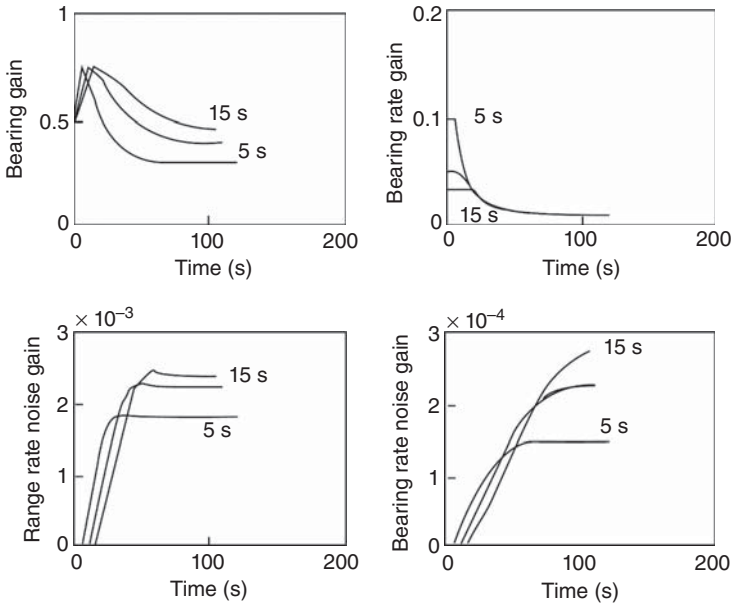


Figure 5.12 Kalman gains.

block diagram description. The continuous-time estimator (Kalman–Bucy filter) is also described.

Three different derivations of the Kalman gain have been given. A fourth derivation—using Newton's method—has been given by Humpherys et al [51].

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 that there is correlation between plant and measurement noise sources and correlated measurement errors were discussed. Relationships between stationary continuous-time and Kalman filter 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 (i) estimating the state (phase and amplitude) of a harmonic oscillator and (ii) a discrete-time Kalman filter implementation of a five-dimensional radar tracking problem.

The discrete-time Kalman filter is recursive, in that it turns old values into new values. It is essentially a recycler of estimates and covariances.

An estimator for the state of a dynamic system at time t, using measurements made *after* time t, is called a *smoother* (next chapter).

There is nothing in the Kalman filter derivation that depends on the underlying probability distributions being Gaussian. Kalman recognized that, assuming only linearity, the mean will remain the least-mean-squared error estimator and second central moment will be the covariance matrix of estimation error. Nothing in the

derivation depends on the higher order moments of the underlying distributions, so long as everything is linear.

5.12.2 Important Equations to Remember

5.12.2.1 Kalman Filter The *discrete-time model* for a linear stochastic system has the form

$$\begin{aligned} x_k &= \Phi_{k-1}x_{k-1} + G_{k-1}w_{k-1}, \\ z_k &= H_k x_k + v_k, \end{aligned}$$

where the zero-mean uncorrelated 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

$$\begin{aligned} \hat{x}_{k(-)} &= \Phi_{k-1}\hat{x}_{(k-1)(+)}, \\ P_{k(-)} &= \Phi_{k-1}P_{(k-1)(+)}\Phi_{k-1}^{\mathrm{T}} + G_{k-1}Q_{k-1}G_{k-1}^{\mathrm{T}}, \\ \hat{x}_{k(+)} &= \hat{x}_{k(-)} + \overline{K}_k(z_k - H_k\hat{x}_{k(-)}), \\ \overline{K}_k &= P_{k(-)}H_k^{\mathrm{T}}(H_kP_{k(-)}H_k^{\mathrm{T}} + R_k)^{-1}, \\ P_{k(+)} &= P_{k(-)} - \overline{K}_kH_kP_{k(-)}, \end{aligned}$$

where the $(-)$ indicates the a priori values of the variables (before the information in the measurement is used) and the $(+)$ indicates the a posteriori values of the variables (after the information in the measurement is used). The variable $\overline{K}$ is the Kalman gain.

5.12.2.2 Kalman–Bucy Filter The *continuous-time model* for a linear stochastic system has the form

$$\begin{aligned} \frac{d}{dt}x(t) &= F(t)x(t) + G(t)w(t), \\ z(t) &= H(t)x(t) + v(t), \end{aligned}$$

where the zero-mean uncorrelated 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, has the form

$$\begin{aligned} \frac{d}{dt}\hat{x}(t) &= F(t)\hat{x}(t) + \overline{K}(t)[z(t) - H(t)\hat{x}(t)], \\ \overline{K}(t) &= P(t)H^{\mathrm{T}}(t)R^{-1}(t), \\ \frac{d}{dt}P(t) &= F(t)P(t) + P(t)F^{\mathrm{T}}(t) - \overline{K}(t)R(t)\overline{K}^{\mathrm{T}}(t) + G(t)Q(t)G^{\mathrm{T}}(t). \end{aligned}$$

PROBLEMS

5.1 A scalar discrete-time random sequence x_k is given by

$$x_{k+1} = 0.5x_k + w_k,$$
$$\mathrm{E}x_0 = 0, \mathrm{E}x_0^2 = 1, \mathrm{E}w_k^2 = 1, \mathrm{E}w_k = 0,$$

where w_k is the white noise. The observation equation is given by

$$z_k = x_k + v_k,$$

$\mathrm{E}v_k = 0, \mathrm{E}v_k^2 = 1$, and v_k is also white noise. The terms x_0, w_k, and v_k are all random. Derive a (nonrecursive) expression for

$$\mathrm{E}[x_2|z_0, z_1, z_2].$$

5.2 For the system given in Problem 5.1:
(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.

5.3 Prove that any Gaussian likelihood function $\mathcal{L}(x,\ \mu_x,\ Y_{xx})$ achieves its maximum at the set of points

$$\underset{x}{\text{argmax}}\, \mathcal{L}(x, \mu_x,\ Y_{xx}) = \left\{ \mu_x + \sum_{\lambda_i=0} a_i e_i \ \mid\ a_i \in \Re \right\},$$

where the summation is over the zero eigenvectors of Y_{xx}. That is, if no eigenvalue of Y_{xx} is zero, the maximum occurs at μ_x. However, if Y_{xx} has zero eigenvalues, the same maximum value is attained at μ_x plus any linear combination of the corresponding zero eigenvectors of Y_{xx}.

5.4 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):

$$\mathrm{E}v_k = \mathrm{E}w(t) = 0,$$

$$\mathrm{E}w(t_1)w(t_2) = 1\delta(t_2 - t_1),$$

$$\mathrm{E}v_{k_1\mathrm{T}}v_{k_2\mathrm{T}} = 1\Delta(k_2 - k_1),$$

$$\mathrm{E}\langle v_k w^{\mathrm{T}}\rangle = 0.$$

Derive the minimum mean-squared-filter of $x(t)$ for all t.

5.5 In Problem 5.3, 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 5.4 can be modified to

$$z_{kT+\eta} = x_{kT+\eta} + v_{kT+\eta},$$

where

$$k = 0, 1, 2, \ldots, 0 \le \eta \le \eta_0.$$

Let $T = 1\mathrm{s}$, $\eta_0 = 0.1$ (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.

5.6 Prove the condition in the discussion following Equation 5.9 that $\mathrm{E}w_k z_i^{\mathrm{T}} = 0$ for $i = 1, \ldots k$ when w_k and v_k are uncorrelated and white.

5.7 In Example 5.8, 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 $P_0, Q, R, \sigma_r^2, \sigma_\theta^2, \sigma_1^2$, and σ_2^2.

5.8 For the same problem as Problem 5.7, obtain values of the plant covariance Q for the four-state model such 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 9.)

5.9 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).

5.10 *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.

5.11 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.

5.12 Show that the matrices P_k and $P(t)$ of Equations 5.23 and 5.116 are *symmetric*. That is, $P_k^{\mathrm{T}} = P_k$ and $P^{\mathrm{T}}(t) = P(t)$.

5.13 Derive the observability matrices for Examples 5.7 and 5.8 to determine whether these systems are observable.

5.14 A vector discrete-time random sequence x_k is given by

$$\begin{aligned} x_k &= \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x_{k-1} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} w_{k-1}, \\ w_k &\sim \mathcal{N}(0, 1) \text{ and white.} \end{aligned}$$

The observation equation is given by

$$\begin{aligned} z_k &= [1 \quad 0] x_k + v_k, \\ v_k &\sim \mathcal{N}[0, 2 + (-1)^k] \text{ and white.} \end{aligned}$$

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

$$P_0 = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}.$$

5.15 Calculate the values of $P_{k(+)}$, $\overline{K}_k$, and $\hat{x}_{k(+)}$ by serial processing of a vector measurement:

$$\hat{x}_{k(-)} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \; P_{k(-)} = \begin{bmatrix} 4 & 1 \\ 1 & 9 \end{bmatrix}, \; H_k = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \; R_k = \begin{bmatrix} 3 & 0 \\ 0 & 4 \end{bmatrix}, \; z_k = \begin{bmatrix} 3 \\ 4 \end{bmatrix}.$$

REFERENCES

[1] B. D. O. Anderson and J. B. Moore, *Optimal Filtering*, Prentice-Hall, Englewood Cliffs, NJ, 1979.

[2] S. M. Bozic, *Digital and Kalman Filtering: An Introduction to Discrete-Time Filtering and Optimal Linear Estimation*, John Wiley & Sons, Inc., New York, 1979.

[3] K. Brammer and G. Siffling, *Kalman–Bucy Filters*, Artech House, Norwood, MA, 1989.

[4] R. G. Brown, *Introduction to Random Signal Analysis and Kalman Filtering*, John Wiley & Sons, Inc., New York, 1983.

[5] A. E. Bryson Jr. and Y.-C. Ho, *Applied Optimal Control*, Blaisdell, Waltham, MA, 1969.

[6] R. S. Bucy and P. D. Joseph, *Filtering for Stochastic Processes with Applications to Guidance*, American Mathematical Society, Chelsea Publishing, Providence, RI, 2005.

[7] D. E. Catlin, *Estimation, Control, and the Discrete Kalman Filter*, Springer-Verlag, New York, 1989.

[8] C. K. Chui and G. Chen, *Kalman Filtering with Real-Time Applications*, Springer-Verlag, New York, 1987.

[9] 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.

[10] A. H. Jazwinski, *Stochastic Processes and Filtering Theory*, Dover Publishers, Mineola, NY, 2009 (reprint of 1970 edition by Academic Press).

[11] T. Kailath, *Lectures on Weiner and Kalman Filtering*, Springer-Verlag, New York, 1981.

[12] P. S. Maybeck, *Stochastic Models, Estimation, and Control*, Vol. 1, Academic Press, New York, 1979.

[13] P. S. Maybeck, *Stochastic Models, Estimation, and Control*, Vol. 2, Academic Press, New York, 1982.

[14] J. M. Mendel, *Lessons in Digital Estimation Techniques*, Prentice-Hall, Englewood Cliffs, NJ, 1987.

[15] J. M. Mendel, "Kalman filtering and other digital estimation techniques," *IEEE Individual Learning Package*, IEEE, New York, 1987.

[16] N. E. Nahi, *Estimation Theory and Applications*, John Wiley & Sons, Inc., New York, 1969; reprinted by Krieger, Melbourne, FL, 1975.

[17] P. A. Ruymgaart and T. T. Soong, *Mathematics of Kalman–Bucy Filtering*, Springer-Verlag, New York, 1988.

[18] H. W. Sorenson, Ed., *Kalman Filtering: Theory and Application*, IEEE Press, New York, 1985.

[19] J. F. Riccati, "Animadversationnes in aequationes differentiales secundi gradus," *Acta Eruditorum Quae Lipside Publicantur Supplementa*, Vol. 8, pp. 66–73, 1724.

[20] S. Bittanti, A. J. Laub, and J. C. Willems, Eds., *The Riccati Equation*, Springer-Verlag, Berlin, 1991.

[21] 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.

[22] L. Dieci, "Numerical integration of the differential Riccati equation and some related issues," *SIAM Journal of Numerical Analysis*, Vol. 29, pp. 781–815, 1992.

[23] J. Aldrich, "R. A. Fisher and the making of maximum likelihood 1912–1922," *Statistical Science*, Vol. 12. No. 3, pp. 162–176, 2004.

[24] A. Hald, *A History of Mathematical Statistics from 1750 to 1930*, John Wiley & Sons, Inc., New York, 1998.

[25] A. Hald, "On the history of maximum likelihood in relation to inverse probability and least squares," *Statistical Science*, Vol. 14, No. 2, pp. 214–222, 1999.

[26] J. W. Pratt, "F. Y. Edgeworth and R. A. Fisher on the efficiency of maximum likelihood estimation," *Annals of Statistics*, Vol. 4, No. 3, pp. 501–514, 1976.

[27] S. M. Stigler, "The epic story of maximum likelihood," *Statistical Science*, Vol. 22, No. 4, pp. 598–620, 2007.

[28] R. E. Kalman, "A new approach to linear filtering and prediction problems," *ASME Journal of Basic Engineering*, Vol. 82, pp. 34–45, 1960.

[29] W. J. Duncan, "Some devices for the solution of large sets of simultaneous linear equations (with an appendix on the reciprocation of partitioned matrices)," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, Series 7*, Vol. 35, pp. 660–670, 1944.

[30] A. Bain and D. Crisan, *Fundamentals of Stochastic Filtering*, Springer, New York, 2009.

[31] J. L. Crassidis and J. L. Junkins, *Optimal Estimation of Dynamic Systems*, Chapman and Hall (CRC Press), Boca Raton, FL, 2004.

[32] H. W. Sorenson, "Kalman filtering techniques," in *Advances in Control Systems* Vol. 3 (C. T. Leondes, Ed.), Academic Press, Baltimore, MD, pp. 219–292, 1966.

[33] 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.

[34] H. Kwaknernaak and R. Sivan, *Linear Optimal Control Systems*, John Wiley & Sons, Inc., New York, 1972.

[35] R. E. Kalman, "New methods in Wiener filtering," in *Proceeding of the First Symposium on Engineering Applications of Random Function Theory and Probability*, J. L. Bogdanoff and F. Kozin, Eds., John Wiley & Sons, Inc., New York, pp. 270–388, 1963.

[36] 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.

[37] D. Slepian, "Estimation of signal parameters in the presence of noise," *IRE Transactions on Information Theory*, Vol. IT - 3, pp. 68–69, 1954.

[38] J. E. Potter, A Matrix Equation Arising in Statistical Estimation Theory, Report No. CR - 270, National Aeronautics and Space Administration, New York, 1965.

[39] A. G. J. MacFarlane, "An eigenvector solution of the optimal linear regulator," *Journal of Electronic Control*, Vol. 14, pp. 643–654, 1963.

[40] J. E. Potter, "Matrix quadratic solutions," *SIAM Journal of Applied Mathematics*, Vol. 14, pp. 496–501, 1966.

[41] A. F. Fath, "Computational aspects of the linear optimal regulator problem," *IEEE Transactions on Automatic Control*, Vol. AC - 14, pp. 547–550, 1969.

[42] S. F. Schmidt, "Applications of state-space methods to navigation problems," in *Advances in Control Systems* (C. T. Leondes, Ed.), Vol. 3, Academic Press, New York, pp. 293–340, 1966.

[43] 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.

[44] M. S. Grewal and R. S. Miyasako, "Gyro compliance estimation in the calibration and alignment of inertial measurement units," in *Proceedings of 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.

[45] 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.

[46] J. H. Blakelock, *Automatic Control of Aircraft and Missiles*, John Wiley & Sons, Inc., New York, 1965.

[47] 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.

[48] 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.

[49] 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, 1977, pp. 1192–1195.

[50] M. Schwartz and L. Shaw, *Signal Processing, Discrete Spectral Analysis, Detection, and Estimation*, McGraw-Hill, New York, 1975.

[51] J. Humpherys, P. Redd, and J. West, "A fresh look at the Kalman filter," *SIAM Review*, Vol. 54, No, 4, pp. 801–823, 2012.

6

OPTIMAL SMOOTHERS

> Many difficulties which nature throws in our way may be smoothed away by the exercise of intelligence.
>
> —Livy [Titus Livius] (59BCE–17CE)

6.1 CHAPTER FOCUS

6.1.1 Smoothing and Smoothers

6.1.1.1 What We Mean by "Optimal Smoothing?" The term *smoothing* has many meanings in different contexts, dating from antiquity. The term *optimal smoothing* has been used from 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 the 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®, Fourth Edition.
Mohinder S. Grewal and Angus P. Andrews.
© 2015 John Wiley & Sons, Inc. Published 2015 by John Wiley & Sons, Inc.

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 the Wiener filtering theory.

6.1.2 Kalman Filtering, Prediction, Interpolation and Smoothing

Kalman filtering theory was developed in Chapters 2 through 5. The same theoretical models have been used in developing optimal smoothing methods.

Prediction was also covered in Chapter 5. 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 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 be 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 is the subject of this chapter. Its relationship to optimal predictors and optimal filters is illustrated in Figure 6.1, 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.
- *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

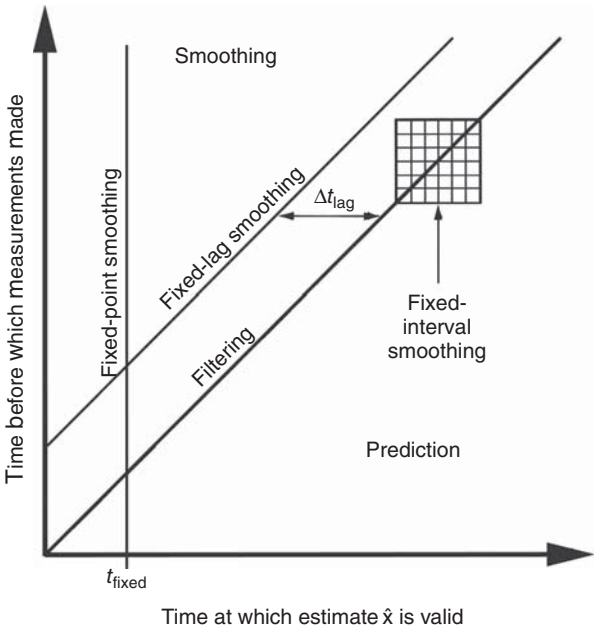


Figure 6.1 Estimate/measurement timing constraints.

is valid. The three different types of smoothers labeled in Figure 6.1 are briefly described in the following subsection.

6.1.3 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 6.1. 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.

6.1.3.1 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 6.1.

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 post-processing 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 6.2.

6.1.3.2 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 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 6.1.

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 6.3.

6.1.3.3 Fixed-point Smoothers Fixed-point smoothers generate an estimate $\hat{\mathbf{x}}(t_{fixed})$ of $\boldsymbol{x}$ 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 6.1. 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. They are used for initial alignment of inertial navigation systems (INS) [1] 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 6.4.

6.1.4 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 actually 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.

6.1.5 Smoothing Applications

In practice, smoothing is often overlooked as a practical method for getting better estimates than those attainable by filtering alone. Depending on attributes of the application, the improvement of smoothing over filtering can be many orders of magnitude in mean-squared estimation uncertainty.

6.1.5.1 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 oversensitivity 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.

6.1.6 Improvement over Filtering

Theoretical limits on the asymptotic improvement of smoothing over filtering were published by Anderson [2] in 1969 for the case that the dynamic system is asymptotically exponentially stable. The limit in that case is a factor of two in mean-squared estimation uncertainty, but greater improvement is attainable for unstable systems.

6.1.6.1 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 $P_{[f]}$ 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.

6.1.6.2 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 6.2 through 6.4 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.

6.1.6.3 The Anderson–Chirarattananon Bound This result, due to Brian D. O. Anderson and Surapong Chirarattananon [3], was derived in the frequency domain and uses 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 across all frequencies. The peak SNR then occurs at that frequency where the signal power spectral density peaks.

The result, shown in Figure 6.2 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 6.2(a) is a lower bound on the improvement ratio; that in Figure 6.2(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.

6.2 FIXED-INTERVAL SMOOTHING

6.2.1 Performance Analysis in Continuous Time

6.2.1.1 Scalar Parametric Models

Linear Time-Invariant Models in Continuous Time Smoother performance models are developed in continuous time for pedagogical reasons: the resulting model

[1] Setting $n = 1$ has more to do with dimensionality of the independent parameters (for plotting) than the feasibility of the performance analysis.

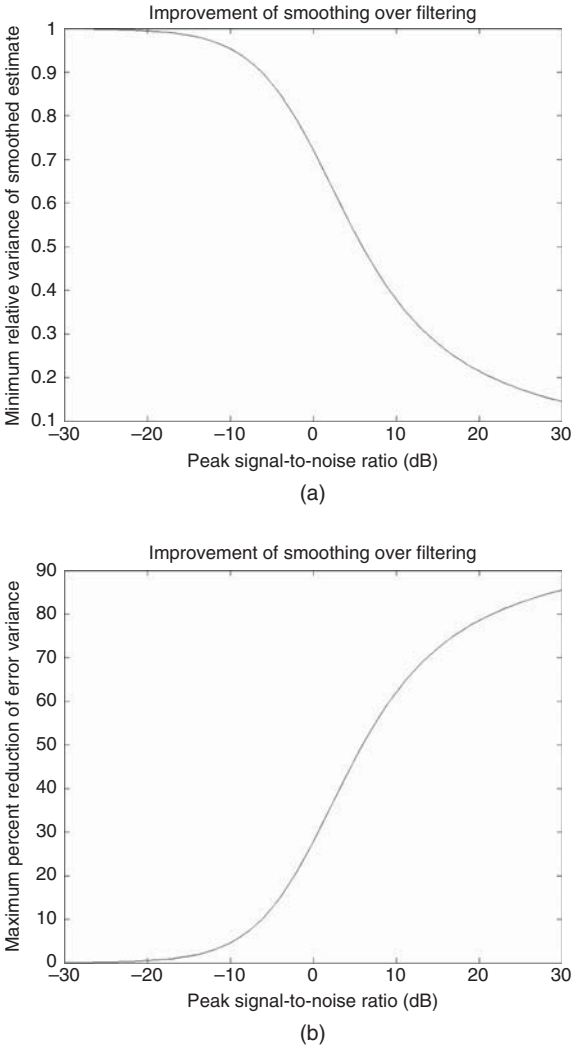


Figure 6.2 Anderson–Chirarattananon bound on smoothing variance improvement over filtering. (a) Relative improvement and (b) percent improvement.

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. They are then demonstrated for the scalar state variable case, although the general approach works for arbitrary state vector and measurement vector dimension.

Stochastic System Model The parametric stochastic system model is

$$\begin{aligned} \dot{x} &= F\,x + w \,(\text{dynamic system model}) \\ z &= H\,x + v \,(\text{measurement model}) \\ \dot{P} &= FP + PF - PHR^{-1}H^{\mathrm{T}}P + Q \,(\text{Riccati differential equation}). \end{aligned} \tag{6.1}$$

Parameter Units for $n = 1$ When the state vector has dimension $n = 1$, the scalar Riccati equation variables and parameters have the following units:

$$\begin{aligned} &P \quad \text{has the same units as } x^2. \\ &F \quad \text{has units of } 1/\text{time (e.g., } \mathrm{s}^{-1}). \\ &H \quad \text{has } z \text{ (measurement) units divided by } x \text{ (state variable) units.} \\ &R \quad \text{has the units of time} \times z^2 \,(\text{measurement units})^2. \\ &Q \quad \text{has the units of } x^2/\text{time}. \end{aligned} \tag{6.2}$$

The seemingly strange units or 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 6.1 changes, which can make the performance of the backward filter model different from that of the forward filter model. The 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 the 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-filter models (forward and backward); the estimates obtained by 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,
$P_{[b]}(t)$ 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

$$\hat{x}_{[s]}(t) = \hat{x}_{[f]}(t) + \underbrace{P_{[f]}(t)[P_{[f]}(t) + P_{[b]}(t)]^{-1}}_{\overline{K}}[\hat{x}_{[b]}(t) - \hat{x}_{[f]}(t)],$$

which is essentially the Kalman filter observational update formula with

$$\begin{aligned}
\hat{x}_{[f]}(t) &= \text{the smoother a priori estimate} \\
H &= I\,(\text{identity matrix}) \\
z(t) &= \hat{x}_{[b]}(t)\,(\text{backward filter estimate}) \\
R &= P_{[b]}(t) \\
\overline{K} &= P_{[f]}(t)[P_{[f]}(t) + P_{[b]}(t)]^{-1}\,(\text{Kalman gain}) \\
\hat{x}_{[s]}(t) &= \text{the smoother a posteriori estimate} \\
P_{[s]}(t) &= \text{the a posteriori covariance of smoother estimation uncertainty} \\
&= P_{[f]}(t) - P_{[f]}(t)[P_{[f]}(t) + P_{[b]}(t)]^{-1}P_{[f]}(t) \\
&= P_{[f]}(t)\{I - [P_{[f]}(t) + P_{[b]}(t)]^{-1}P_{[f]}(t)\} \\
&= P_{[f]}(t)[P_{[f]}(t) + P_{[b]}(t)]^{-1}\{[P_{[f]}(t) + P_{[b]}(t)] - P_{[f]}(t)\} \\
&= P_{[f]}(t)[P_{[f]}(t) + P_{[b]}(t)]^{-1}P_{[b]}(t). \qquad (6.3)
\end{aligned}$$

The last of these equations characterizes smoother covariance as a function of forward and backward filter covariances, which are the solutions of the appropriate Riccati equations.

6.2.1.2 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 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 6.3 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:

$$P_{[f]} = \sqrt{\left(\frac{FR}{H^2}\right)^2 + \left(\frac{QR}{H^2}\right)} + \left(\frac{FR}{H^2}\right) \tag{6.4}$$

$$P_{[b]} = \sqrt{\left(\frac{FR}{H^2}\right)^2 + \left(\frac{QR}{H^2}\right)} - \left(\frac{FR}{H^2}\right) \tag{6.5}$$

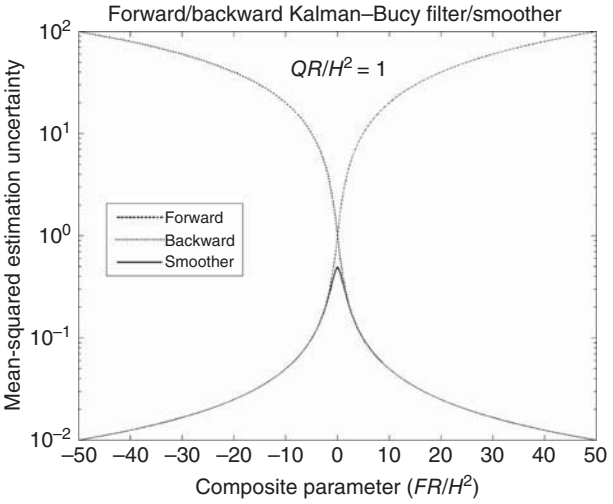


Figure 6.3 Dependence of Kalman–Bucy filter and smoother performance on FR/H^2.

$$P_{[s]} = \frac{\left(\frac{QR}{H^2}\right)}{2\sqrt{\left(\frac{FR}{H^2}\right)^2 + \left(\frac{QR}{H^2}\right)}}. \tag{6.6}$$

All these formulas depend on just two parameter expressions:

$$\left(\frac{FR}{H^2}\right) \text{ and } \left(\frac{QR}{H^2}\right),$$

meaning that mean-squared filter and smoother performance can be plotted as a function of these two parameter expressions.

Results Figures 6.3 and 6.4 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 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 6.3, 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 for 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 Figure 6.4 (a) and (b), which are 3D 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 6.4(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 [4], which assumed the dynamic system to have bounded signal power.

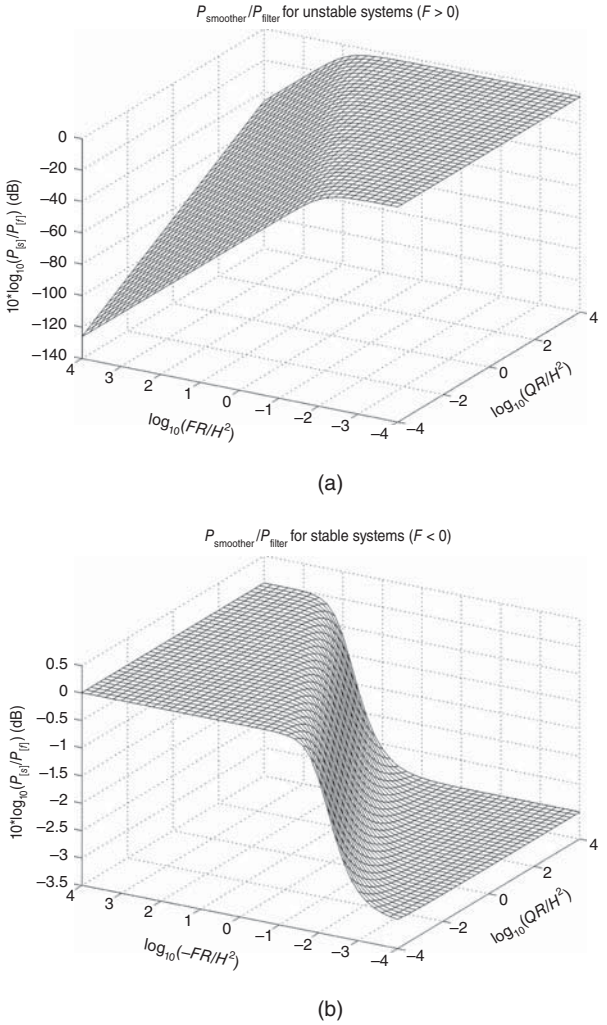


Figure 6.4 Smoothing improvement ratio versus FR/H^2 and QR/H^2. (a) Unstable system and (b) stable system.

Random Walk Conditions The smoothing improvement ratio $P_{[s]}/P_{[f]} \to 1/2$ (−3 dB) as $(FR/H^2) \to 0$, the random walk model. That is, the improvement of smoothing over filtering for a *random walk* is a factor of two decrease in mean-squared estimation uncertainty—or a factor of $\sqrt{2}$ decrease in root-mean-square (RMS) uncertainty.

Stable Systems For $F < 0$ (Figure 6.4 (b)), the smoothing improvement ratio $P_{[s]}/P_{[f]} \to 1/2$ as $(FR/H^2) \to 0$, the random walk model. For $F < 0$, $P_{[s]}/P_{[f]} \to 1/2$ as $(QR/H^2) \to +\infty$, 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 [4].

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.

6.2.1.3 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

$$P_{[f]}(t) = A_{[f]}(t)B_{[f]}^{-1}(t) \tag{6.7}$$

$$\begin{bmatrix} A_{[f]}(t) \\ B_{[f]}(t) \end{bmatrix} = \exp\left(\left(t - t_{\text{start}}\right) \begin{bmatrix} F & Q \\ H^2/R & -F \end{bmatrix} \right) \begin{bmatrix} A_{[f]}(t_{\text{start}}) \\ B_{[f]}(t_{\text{start}}) \end{bmatrix} \tag{6.8}$$

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

$$\tau = \frac{R}{\sqrt{R(F^2R + H^2Q)}} \tag{6.9}$$

$$P_{[f]}(t) = \frac{\sqrt{R(F^2R + H^2Q)}}{H^2} \frac{(e^{(t-t_{\text{start}})/\tau} + e^{-(t-t_{\text{start}})/\tau})}{(e^{(t-t_{\text{start}})/\tau} - e^{-(t-t_{\text{start}})/\tau})} + \frac{FR}{H^2}. \tag{6.10}$$

For the backward filter, this Riccati equation solution is

$$P_{[b]}(t) = A_{[b]}(t)B_{[b]}^{-1}(t) \tag{6.11}$$

$$\begin{bmatrix} A_{[b]}(t) \\ B_{[b]}(t) \end{bmatrix} = \exp\left(\left(t - t_{\text{end}}\right) \begin{bmatrix} -F & Q \\ H^2/R & F \end{bmatrix} \right) \begin{bmatrix} A_{[b]}(t_{\text{end}}) \\ B_{[b]}(t_{\text{end}}) \end{bmatrix}, \tag{6.12}$$

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

$$P_{[b]}(t) = \frac{\sqrt{R(F^2R + H^2Q)}}{H^2} \frac{(e^{(t_{\text{end}}-t)/\tau} + e^{-(t_{\text{end}}-t)/\tau})}{(e^{(t_{\text{end}}-t)/\tau} - e^{-(t_{\text{end}}-t)/\tau})} - \frac{FR}{H^2}. \tag{6.13}$$

The plot in Figure 6.5 was generated using these formulas in the MATLAB® m-file `FiniteFIS.m` on the companion Wiley web site. Performance is flat in the mid-section

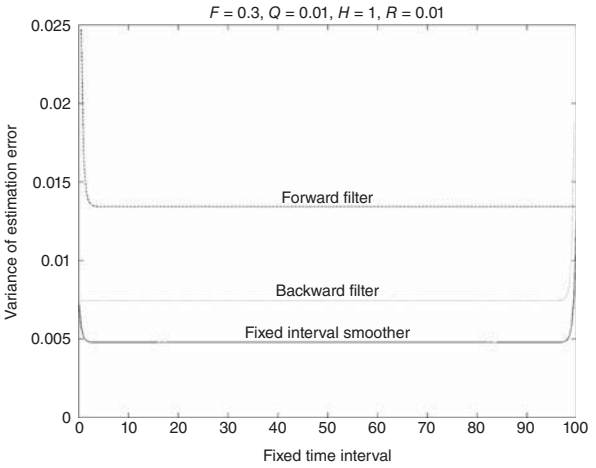


Figure 6.5 Fixed-interval smoothing on finite interval.

because the model is time invariant. The plot shows the "end effects" at both the 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.

6.2.2 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 on Section 6.2.1.3, 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 6.6, 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 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:

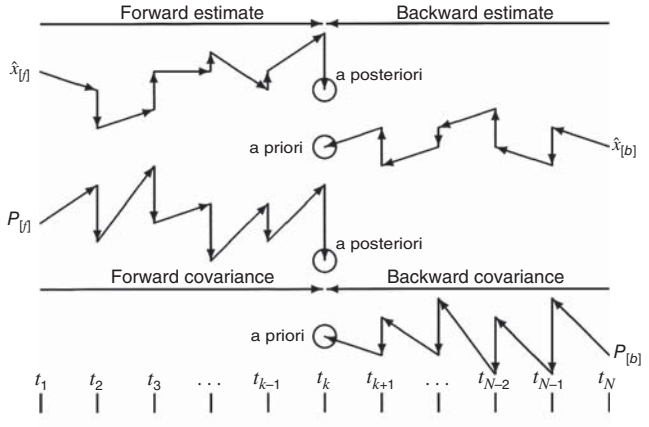


Figure 6.6 Forward and backward filter outputs.

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 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,

$$\hat{x}_{[b],k-1(-)} = \Phi^{-1}_{[f],k-1}\hat{x}_{[b],k(+)}\,(\text{predictor}) \tag{6.14}$$

$$P_{[b],k-1(-)} = \Phi^{-1}_{[f],k-1}P_{[b],k(+)}\Phi^{-\mathrm{T}}_{[f],k-1} + Q_k \tag{6.15}$$

(the subscript "[b]" is 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 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:

$$\hat{x}_{[s],k(+)} = \hat{x}_{[f],k(+)} + \underbrace{P_{[f],k(+)}[P_{[f],k(+)} + P_{[b],k(-)}]^{-1}}_{\overline{K}}[\hat{x}_{[b],k(-)} - \hat{x}_{[f],k(+)}], \tag{6.16}$$

where $\hat{x}_{s,k}(+)$ is the smoothed estimate.
One can also compute the covariance of smoother uncertainty,

$$P_{[s],k} = \overline{K}P_{[b],k(-)}. \tag{6.17}$$

It is not required in the smoother operation, but it is useful for understanding how good the smoothed estimate is supposed to be.

6.2.2.1 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 6.6. This is 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.

6.2.2.2 Optimal Combination of the Two Filter Outputs Equation 6.16 has the form of the 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),
$\overline{K} = P_{[f],k(+)}[P_{[f],k(+)} + P_{[b],k(-)}]^{-1}$ is the effective Kalman gain, and
$\hat{x}_{[s],k(+)}$ is the resulting optimal smoother estimate.

Computational issues Equation 6.16 requires the inverse of the covariance matrix P from the forward pass, and Equations 6.14 and 6.15 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 Wiley web site 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. A sample output is shown in Figure 6.7. As an empirical measure of smoothing performance relative to the performance of the two filter, the 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 theoretical basis for

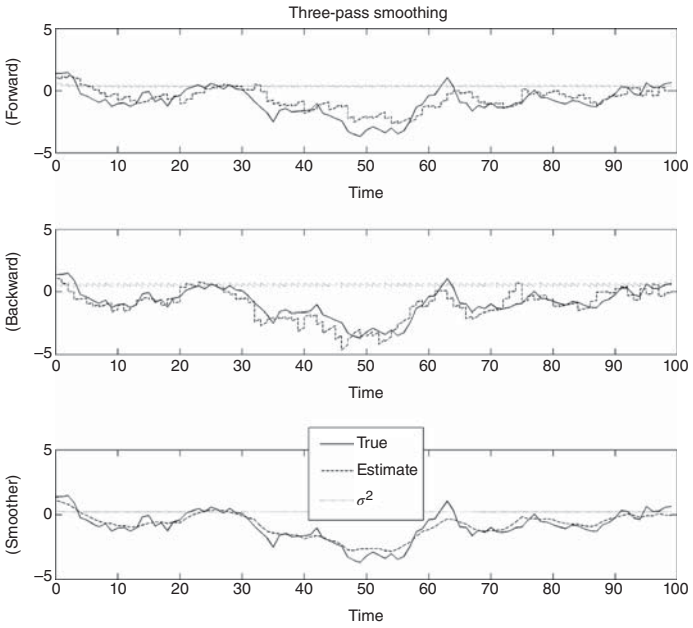


Figure 6.7 Simulated three-pass fixed-interval smoother.

optimal smoothing, to show that a relatively straightforward implementation method exists, and to derive the following example of smoothing performance for very simple models.

6.2.3 Rauch–Tung–Striebel (RIS) Two-pass Smoother

This fixed-interval two-pass implementation is the fastest fixed-interval smoother [5], and it has been used quite successfully for decades. The algorithm was published by Herbert E. Rauch, F. Tung, and Charlotte T. Striebel in 1965 [6], 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

$$\hat{x}_{[s]N} = \hat{x}_{N(+)}, \tag{6.18}$$

then computed recursively by the formulas

$$\hat{x}_{[s]k} = \hat{x}_{k(+)} + A_k(\hat{x}_{[s]k+1} - \hat{x}_{k+1(-)}), \tag{6.19}$$

$$A_k = P_{k(+)}\Phi_k^{\mathrm{T}} P_{k+1(-)}^{-1}. \tag{6.20}$$

6.2.3.1 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.

6.2.3.2 Performance The covariance of uncertainty of the smoothed estimate can also be computed on the second pass:

$$P_{[s]k} = P_{k(+)} + A_k(P_{[s]k+1} - P_{k+1(-)})A_k^{\mathrm{T}}, \tag{6.21}$$

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.

6.3 FIXED-LAG SMOOTHING

6.3.1 Stability Problems of Early Methods

Rauch [7] published the earliest fixed-lag smoothing method based on the Kalman filter model in 1963. Kailath and Frost [8] later discovered that the Kalman filter and the Rauch fixed-lag smoother are an "adjoint pair," and Kelly and Anderson [9] showed that 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 [10] first introduced the idea of augmenting the state vector to recast the smoothing problem as a filtering problem. Biswas and Mahalanabis also proved stability of their fixed-lag smoother implementation [11] and determined its computational requirements [12].

We present in Section 6.3.3 an augmented state vector implementation from Biswas and Mahalanabis [10], Premier and Vacroux [13], and Moore [14], which has become standard. Prasad and Mahalanabis [15] and Tam and Moore [16] 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 "generic" fixed-lag smoothing performs relative to filtering and how relative performance depends on the lag time and other parameters of the application.

6.3.2 Performance Analysis

6.3.2.1 Analytical Model in Continuous Time We can use the same scalar Kalman–Bucy filter model of Equations 6.1 and 6.2 from Section 6.2.1 to characterize the relative improvement of fixed-lag smoothing over filtering as a function of lag time and the parameters of the filter model.

How this model is used for the fixed-lag smoother is illustrated in Figure 6.8. At any time t, the effect of the lag time Δt_{lag} on the estimate at time t, given measurements

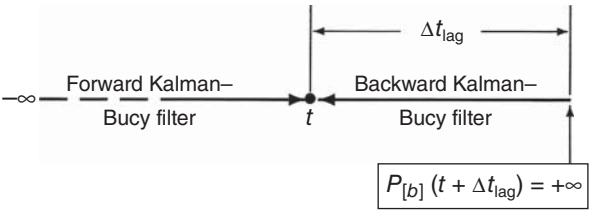


Figure 6.8 Kalman–Bucy fixed-lag smoother model.

out to time $t + \Delta t_{\text{lag}}$, is modeled by an additional filter operating backward from time $t + \Delta t_{\text{lag}}$ to time t, starting with no information at all at time $t + \Delta t_{\text{lag}}$ (i.e., $P(t + \Delta t_{\text{lag}}) = +\infty$). The improvement of fixed-lag smoothing over filtering can then be evaluated as it was in Section 6.2.1, except that this example has as an additional parameter the lag time Δt_{lag}.

6.3.2.2 Forward Kalman–Bucy Filter Performance In Section 6.2.1, the Kalman–Bucy forward filter performance at steady state ($\dot{P}_{[f]} = 0$) was shown to be

$$P_{[f]}(t) = \sqrt{\left(\frac{FR}{H^2}\right)^2 + \left(\frac{QR}{H^2}\right)} + \left(\frac{FR}{H^2}\right), \tag{6.22}$$

which is a formula for filtering performance as a function of the two composite parameters (FR/H^2) and (QR/H^2).

6.3.2.3 Backward Kalman–Bucy Filter Performance For the nonsteady-state problem of the backward filter, we can use the linearized Riccati equation solution (originally published by Jacopo Riccati in 1724 [17]) from Section 5.8.3. The solution for the backward filter covariance $P_{[b]}(t)$ at time t, having started at time $t + \Delta t_{\text{lag}}$ with initial value $P_{[b]}(t + \Delta t_{\text{lag}}) = +\infty$ (i.e., with no initial information, modeled by setting $P(0) = 1$ and the vector component below it is equal to 0 in Equation 5.70) is

$$P_{[b]}(t) = \sqrt{\left(\frac{FR}{H^2}\right)^2 + \left(\frac{QR}{H^2}\right)} \frac{e^{\rho} + e^{-\rho}}{e^{\rho} - e^{-\rho}} - \left(\frac{FR}{H^2}\right) \tag{6.23}$$

$$\rho = \left(\frac{H^2 \Delta t_{\text{lag}}}{R}\right) \sqrt{\left(\frac{FR}{H^2}\right)^2 + \left(\frac{QR}{H^2}\right)}, \tag{6.24}$$

which introduces a third composite parameter, $(H^2 \Delta t_{\text{lag}}/R)$.

6.3.2.4 Kalman–Bucy Fixed-lag Smoother Performance The formula for two-filter smoothing (from Section 6.2.1) 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 $(H^2\Delta t_{\text{lag}}/R)$.

6.3.2.5 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 a paper as a function of three parameters, as the infinite-interval smoother performance was plotted as a function of two parameters in Section 6.2.1, 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

$$\begin{array}{llll} \text{as} & \Delta t_{\text{lag}} & \to & 0, \\ \text{as} & \Delta t_{\text{lag}} & \to & +\infty, \end{array} \qquad \begin{array}{rcl} \frac{P_{[s]}(t)}{P_{[f]}(t)} & \to & 1 \text{ (no improvement)}; \\ \rho & \to & +\infty, \\ e^{-\rho} & \to & 0, \\ e^{\rho} & \to & +\infty, \\ \frac{e^{\rho}+e^{-\rho}}{e^{\rho}-e^{-\rho}} & \to & 1, \\ \frac{P_{[s]}(t)}{P_{[f]}(t)} & \to & \dfrac{\left(\frac{QR}{H^2}\right)}{2\left(\frac{FR}{H^2}\right)^2+2\left(\frac{QR}{H^2}\right)+2\left(\frac{FR}{H^2}\right)\sqrt{\left(\frac{FR}{H^2}\right)^2+\left(\frac{QR}{H^2}\right)}}, \end{array}$$

the improvement ratio of the fixed infinite-interval Kalman–Bucy smoother of Section 6.2.1

6.3.3 Biswas–Mahalanabis Fixed-lag Smoother (BMFLS)

The earliest implementations for a fixed-lag smoother were mathematically correct but were found to be poorly condition for numerical implementation. We demonstrate here the relative numerical stability of an implementation published by Moore [14], based on an approach by Premier and Vacroux [13], which is the "state augmentation" filtering approach published earlier by Biswas and Mahalanabis [10]. It is essentially the 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 6.9. If

$$\Delta t_{\text{lag}} = \ell\, \Delta t$$

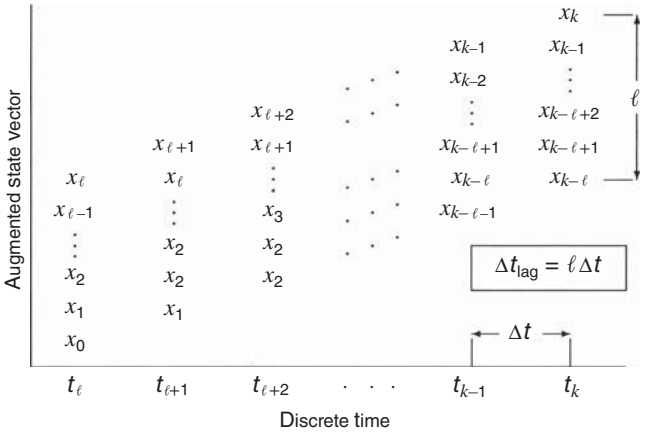


Figure 6.9 Biswas–Mahalanabis augmented state vector.

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 (ℓ_1) n-dimensional subvectors x_k, x_{k-1}, x_{k-2}, … , $x_{k-\ell}$—as shown in Figure 6.9.

6.3.3.1 State Vector In order to distinguish between the Kalman filter state vector and the BMFLS fixed-lag smoother state vector, we will use the following notations:

- $\mathbf{x}_{k,\mathrm{KF}}$ to denote the true value of the state vector of Kalman filter at the kth epoch in discrete time.
- $\hat{\mathbf{x}}_{k,\mathrm{KF}}$ to denote the estimated value of the state vector of Kalman filter at the kth epoch in discrete time.
- $\mathbf{x}_{k,\mathrm{BMFLS}}$ to denote the true value of the state vector of the Biswas–Mahalanabis fixed-lag smoother (BMFLS) at the kth epoch in discrete time.
- $\hat{\mathbf{x}}_{k,\mathrm{BMFLS}}$ to denote the estimated value of the state vector of the BMFLS at the kth epoch in discrete time.

The BMFLS uses $\ell + 1$ lagged values $\mathbf{x}_{k,\mathrm{KF}}$, $\mathbf{x}_{k-1,\mathrm{KF}}$, $\mathbf{x}_{k-2,\mathrm{KF}}$, … , $\mathbf{x}_{k-\ell,\mathrm{KF}}$ 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

$$\mathbf{x}_{k,\mathrm{BMFLS}} \overset{\mathrm{def}}{=} \begin{bmatrix} \mathbf{x}_{k,\mathrm{KF}} \\ \mathbf{x}_{k-1,\mathrm{KF}} \\ \mathbf{x}_{k-2,\mathrm{KF}} \\ \vdots \\ \mathbf{x}_{k-\ell,\mathrm{KF}} \end{bmatrix}. \tag{6.25}$$

The BMFLS is the 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 $\hat{\mathbf{x}}_{k,\text{BMFLS}}$ corresponding to $\mathbf{x}_{k-\ell,\text{KF}}$ represents the smoothed estimate of $\mathbf{x}_{k-\ell}$ using the measurements $\mathbf{z}_j$ for $j \leq k$. That is, the resulting BMFLS estimates the *smoothed* values of

$$\mathbf{x}_{k-\ell,\text{KF}},\ \mathbf{x}_{k-\ell+1,\text{KF}},\ \mathbf{x}_{k-\ell+2,\text{KF}},\ \cdots\ ,\mathbf{x}_{k-1,\text{KF}}$$

plus the *filtered* value $\mathbf{x}_{k,\text{KF}}$, given the measurements

$$\cdots\ ,\ \mathbf{z}_{k-\ell},\ \mathbf{z}_{k-\ell+1},\ \mathbf{z}_{k-\ell+2},\ \cdots\ ,\mathbf{z}_k.$$

In other words, the estimated state vector of the BMFLS will be equivalent to

$$\hat{\mathbf{x}}_{k,\text{BMFLS}} \equiv \begin{bmatrix} \hat{\mathbf{x}}_{k|k} \\ \hat{\mathbf{x}}_{k-1|k} \\ \hat{\mathbf{x}}_{k-2|k} \\ \vdots \\ \hat{\mathbf{x}}_{k-\ell|k} \end{bmatrix}, \tag{6.26}$$

where $\hat{\mathbf{x}}_{k-\ell|k}$ is the notation for the smoothed estimate of $\mathbf{x}_{k-\ell}$ given all measurements up to $\mathbf{z}_k$.

6.3.3.2 State-Transition Matrix If the Kalman filter model has state-transition matrices $\boldsymbol{\Phi}_{k,\text{KF}}$, then the BMFLS state vector has the property that

$$\mathbf{x}_{k+1,\text{BMFLS}} = \begin{bmatrix} \mathbf{x}_{k+1,\text{KF}} \\ \mathbf{x}_{k,\text{KF}} \\ \mathbf{x}_{k-1,\text{KF}} \\ \vdots \\ \mathbf{x}_{k-\ell+2,\text{KF}} \\ \mathbf{x}_{k-\ell+1,\text{KF}} \end{bmatrix} \tag{6.27}$$

$$= \begin{bmatrix} \boldsymbol{\Phi}_{k,\text{KF}} & 0 & 0 & \cdots & 0 & 0 \\ \mathbf{I} & 0 & 0 & \cdots & 0 & 0 \\ 0 & \mathbf{I} & 0 & \cdots & 0 & 0 \\ 0 & 0 & \mathbf{I} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \mathbf{I} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x}_{k,\text{KF}} \\ \mathbf{x}_{k-1,\text{KF}} \\ \mathbf{x}_{k-2,\text{KF}} \\ \vdots \\ \mathbf{x}_{k-\ell+1,\text{KF}} \\ \mathbf{x}_{k-\ell,\text{KF}} \end{bmatrix}. \tag{6.28}$$

That is, the equivalent state-transition matrix for the BMFLS is

$$\boldsymbol{\Phi}_{k,\text{BMFLS}} = \begin{bmatrix} \boldsymbol{\Phi}_{k,\text{KF}} & 0 & 0 & \cdots & 0 & 0 \\ \mathbf{I} & 0 & 0 & \cdots & 0 & 0 \\ 0 & \mathbf{I} & 0 & \cdots & 0 & 0 \\ 0 & 0 & \mathbf{I} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \mathbf{I} & 0 \end{bmatrix}. \tag{6.29}$$

6.3.3.3 Process Noise Covariance If $\mathbf{Q}_{k,\mathrm{KF}}$ is the process noise covariance for the Kalman filter model, then

$$\mathbf{Q}_{k,\mathrm{BMFLS}} = \begin{bmatrix} \mathbf{Q}_{k,\mathrm{KF}} & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}. \tag{6.30}$$

6.3.3.4 Measurement Sensitivity Matrix Because the measurement $\mathbf{z}_k$ is sensitive only to the subvector $\mathbf{x}_{k,\mathrm{KF}}$, the appropriate measurement sensitivity matrix for the BMFLS will be

$$\mathbf{H}_{k,\mathrm{BMFLS}} = \begin{bmatrix} \mathbf{H}_{k,\mathrm{KF}} & 0 & 0 & \cdots & 0 \end{bmatrix}, \tag{6.31}$$

where $\mathbf{H}_{k,\mathrm{KF}}$ is the measurement sensitivity matrix from the Kalman filter model.

6.3.3.5 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

$$\mathbf{R}_{k,\mathrm{BMFLS}} = \mathbf{R}_{k,\mathrm{KF}}, \tag{6.32}$$

the same as for the Kalman filter.

6.3.3.6 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, $1 \le k \le \ell + 1$. 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 6.10, 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

$$\text{length } [\hat{x}_{[s]}(t_k)] = \begin{cases} nk, & 1 \le k \le \ell + 1 \\ n\,(\ell + 1)\,, & k \ge \ell + 1, \end{cases} \tag{6.33}$$

and the associated covariance matrix of smoother state vector uncertainty will grow as $(nk \times nk)$ until $k = \ell + 1$.

Covariance Matrix Augmentation The corresponding buildup of smoother covariance matrices is shown in Table 6.1—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 an 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 Professor Thomas Kailath at the Stanford University, who discovered that the rank of the difference between the original and displacement matrices (called *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:

$$\begin{aligned}
\mathbf{K}_{k,\text{BMFLS}} &= \mathbf{P}_{k,\text{BMFLS}}\mathbf{H}^{\mathrm{T}}_{k,\text{BMFLS}}(\mathbf{H}_{k,\text{BMFLS}}\mathbf{P}_{k,\text{BMFLS}}\mathbf{H}^{\mathrm{T}}_{k,\text{BMFLS}} + \mathbf{R}_k)^{-1} && \text{Gain matrix} \\
\hat{\mathbf{x}}_{k,\text{BMFLS}} &\leftarrow \hat{\mathbf{x}}_{k,\text{BMFLS}} + \mathbf{K}_{k,\text{BMFLS}}(\mathbf{z}_k - \mathbf{H}_{k,\text{BMFLS}}\hat{\mathbf{x}}_{k,\text{BMFLS}}) && \text{Observational state update} \\
\mathbf{P}_{k,\text{BMFLS}} &\leftarrow \mathbf{P}_{k,\text{BMFLS}} - \mathbf{K}_{k,\text{BMFLS}}\mathbf{H}_{k,\text{BMFLS}}\mathbf{P}_{k,\text{BMFLS}} && \text{Observational covariance update} \\
\hat{\mathbf{x}}_{k+1,\text{BMFLS}} &\leftarrow \mathbf{\Phi}_{k,\text{BMFLS}}\hat{\mathbf{x}}_{k,\text{BMFLS}} && \text{Temporal state update} \\
\hat{\mathbf{P}}_{k+1,\text{BMFLS}} &\leftarrow \mathbf{\Phi}_{k,\text{BMFLS}}\hat{\mathbf{P}}_{k,\text{BMFLS}}\mathbf{\Phi}^{\mathrm{T}}_{k,\text{BMFLS}} + \mathbf{Q}_{k,\text{BMFLS}} && \text{Temporal covariance update,}
\end{aligned}$$

where $\mathbf{P}_{k,\text{BMFLS}}$ is the covariance matrix of smoother estimation uncertainty. In the following application to exponentially correlated noise, an initial value for $\mathbf{P}_{k,\text{BMFLS}}$ is determined from the steady-state value of the Riccati equation without measurements.

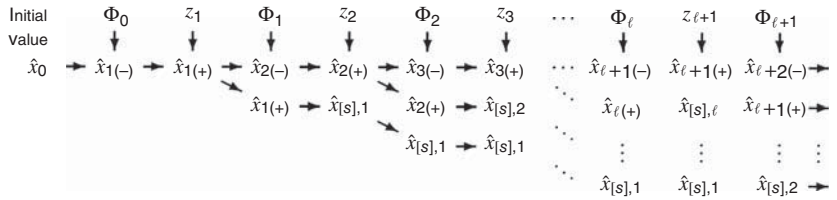


Figure 6.10 Initial state vector transition for BMFLS.

MATLAB Implementation The MATLAB function `BMFLS.m` on the companion Wiley web site implements the complete BMFLS, from the first measurement, through the transition to steady-state state augmentation, and after that.

Example 6.1 (MATLAB Example Using BMFLS.m) Figure 6.11 is a plot of a sample application of `BMFLS.m` using simulated data. The model in this case us 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 measurement vector both have two components, and the signal model is

$$\Phi = \exp(-\Delta t/\tau)\begin{bmatrix} \cos(\omega\,\Delta t) & \sin(\omega\,\Delta t) \\ -\sin(\omega\,\Delta t) & \cos(\omega\,\Delta t) \end{bmatrix} \tag{6.34}$$

$$x_k = \Phi x_{k-1} + w_{k-1} \tag{6.35}$$

TABLE 6.1 BMFLS Fixed-Lag Smoother Covariance Transition

$P_{[s],k}$	A Priori Value	A Posteriori Value $\langle$dim.$\rangle$
$P_{[s],1}$	$P_{[f],1(-)} = \Phi_0 P_0 \Phi_0^{\mathrm{T}} + Q_0$	$P_{[f],1(+)}\ \langle n \times n \rangle$
$P_{[s],2}$	$\begin{bmatrix} \Phi_1 P_{[f],1(+)} \Phi_1^{\mathrm{T}} + Q_1 & \Phi_1 P_{[f],1(+)} \\ P_{[f],1(+)} \Phi_1^{\mathrm{T}} & P_{[f],1(+)} \end{bmatrix}$	$\begin{bmatrix} P_{[f],2(+)} & P_{[s],1,2(+)} \\ P_{[s],1,2(+)}^{\mathrm{T}} & P_{[s],2,2(+)} \end{bmatrix}$
$P_{[s],3}$	$\begin{bmatrix} \Phi_2 P_{[f],2(+)} \Phi_2^{\mathrm{T}} + Q_2 & \Phi_2 P_{[f],2(+)} & \Phi_2 P_{[s],1,2(+)} \\ P_{[f],2(+)} \Phi_2^{\mathrm{T}} & P_{[f],2(+)} & P_{[s],1,2(+)} \\ P_{[s],1,2(+)}^{\mathrm{T}} \Phi_2^{\mathrm{T}} & P_{[s],1,2(+)}^{\mathrm{T}} & P_{[s],2,2(+)} \end{bmatrix}$	$\vdots$
$\vdots$	$\vdots$	$\vdots$
$P_{[s],\ell}$	$\vdots$	$P_{[s](+)}(t_\ell)\ \langle n\ell \times n\ell \rangle$
$P_{[s],\ell+1}$	$\left[\begin{array}{c\|c} \Phi_{\ell-1} P_{[f],\ell-1(+)} \Phi_{\ell-1}^{\mathrm{T}} & \cdots \\ \hline \vdots & P_{[s](+)}(t_\ell) \end{array}\right]$	$P_{[s](+)}(t_{\ell+1})$ $\langle n(\ell+1) \times n(\ell+1) \rangle$
$P_{[s],\ell+2}$	$\left[\begin{array}{c\|c} \Phi_\ell P_{[f],\ell(+)} \Phi_\ell^{\mathrm{T}} & \cdots \\ \hline \cdot & \text{upper-left } \langle n\ell \times n\ell \rangle \\ \cdot & \text{submatrix of} \\ \cdot & P_{[s](+)}(t_{\ell+1}) \end{array}\right]$	$P_{[s](+)}(t_{\ell+2})$ $\langle n(\ell+1) \times n(\ell+1) \rangle$

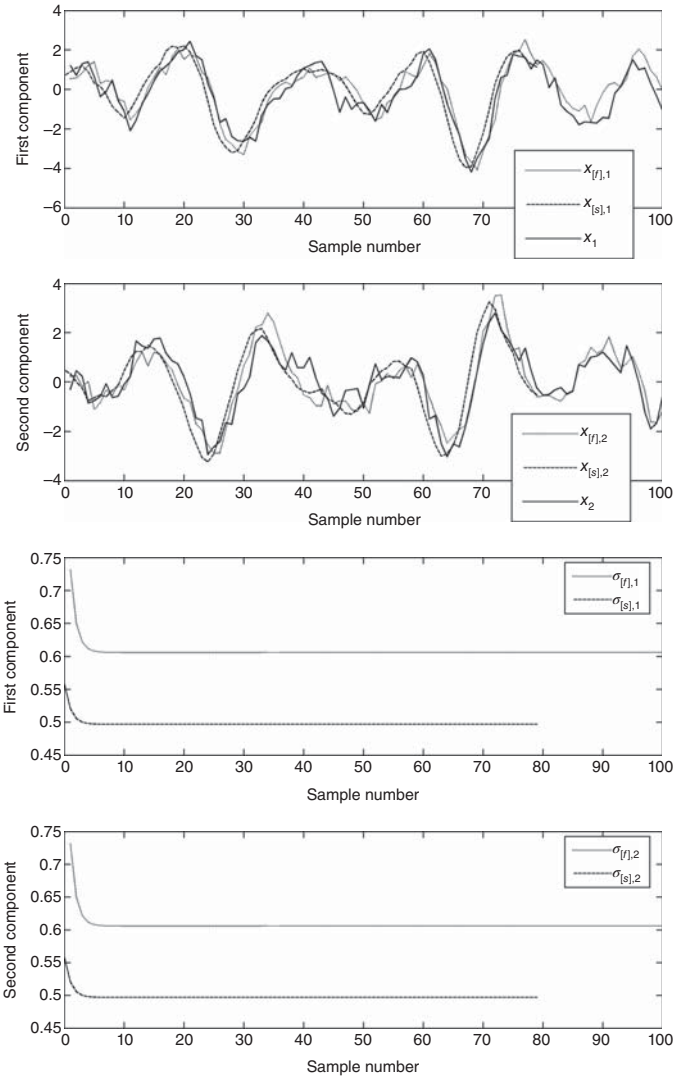


Figure 6.11 Example application of BMFLS.m.

$$z_k = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} x_k + v_k, \tag{6.36}$$

with other parameters as specified in Problem 6.8. The plots include both phase components, and 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 6.2 (Algebraic Riccati Equation Solution) The algebraic Riccati equation for the BMFLS with scalar state vector has a relatively regular structure, from which it is possible to obtain a closed-form solution.

The smoother state-transition matrix in this case has the form

$$\Phi_{[s]} = \begin{bmatrix} \phi & 0 & 0 & \cdots & 0 & 0 \\ 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \tag{6.37}$$

$$\phi = \exp\ (\Delta t/\tau), \tag{6.38}$$

where ϕ is a scalar.

Let the a posteriori smoother covariance matrix

$$P_{[s],\ k-1(+)} = \begin{bmatrix} p_{1,1} & p_{1,2} & p_{1,3} & \cdots & p_{1,\ell} & p_{1,\ell+1} \\ p_{1,2} & p_{2,2} & p_{2,3} & \cdots & p_{2,\ell} & p_{2,\ell+1} \\ p_{1,3} & p_{2,3} & p_{3,3} & \cdots & p_{3,\ell} & p_{3,\ell+1} \\ p_{1,4} & p_{2,4} & p_{3,4} & \cdots & p_{3,\ell} & p_{3,\ell+1} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ p_{1,\ell} & p_{2,\ell} & p_{3,\ell} & \cdots & p_{\ell,\ell} & p_{\ell,\ell+1} \\ p_{1,\ell+1} & p_{2,\ell+1} & p_{3,\ell+1} & \cdots & p_{\ell,\ell+1} & p_{\ell+1,\ell+1} \end{bmatrix}, \tag{6.39}$$

so that the matrix product

$$\begin{aligned} &\Phi_{[s]} P_{[s],\ k-1(+)} \\ &\quad = \begin{bmatrix} \phi p_{1,1} & \phi p_{1,2} & \phi p_{1,3} & \cdots & \phi p_{1,\ell} & \phi p_{1,\ell+1} \\ p_{1,1} & p_{1,2} & p_{1,3} & \cdots & p_{1,\ell} & p_{1,\ell+1} \\ p_{1,2} & p_{2,2} & p_{2,3} & \cdots & p_{2,\ell} & p_{2,\ell+1} \\ p_{1,3} & p_{2,3} & p_{3,3} & \cdots & p_{3,\ell} & p_{3,\ell+1} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ p_{1,\ell-1} & p_{2,\ell-1} & p_{3,\ell-1} & \cdots & p_{\ell-1,\ell} & p_{\ell-1,\ell+1} \\ p_{1,\ell} & p_{2,\ell} & p_{3,\ell} & \cdots & p_{\ell,\ell} & p_{\ell,\ell+1} \end{bmatrix} \end{aligned} \tag{6.40}$$

and the a priori smoother covariance matrix

$$P_{[s],\ k(-)} = \Phi_{[s]} P_{[s],\ k-1(+)} \Phi_{[s]}^{\mathrm{T}} + Q_{[s]} \tag{6.41}$$

$$= \begin{bmatrix} \phi^2 p_{1,1} + q & \phi p_{1,1} & \phi p_{1,2} & \cdots & \phi p_{1,\ell-1} & \phi p_{1,\ell} \\ \phi p_{1,1} & p_{1,1} & p_{1,2} & \cdots & p_{1,\ell-1} & p_{1,\ell} \\ \phi p_{1,2} & p_{1,2} & p_{2,2} & \cdots & p_{2,\ell-1} & p_{2,\ell} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \phi p_{1,\ell-1} & p_{1,\ell-1} & p_{2,\ell-1} & \cdots & p_{\ell-1,\ell-1} & p_{\ell-1,\ell} \\ \phi p_{1,\ell} & p_{1,\ell} & p_{2,\ell} & \cdots & p_{\ell-1,\ell} & p_{\ell,\ell} \end{bmatrix}. \tag{6.42}$$

The measurement sensitivity matrix

$$H_{[s]} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \end{bmatrix}, \tag{6.43}$$

and the matrix product

$$P_{[s],\,k}(-)H_{[s]}^{\mathrm{T}} \tag{6.44}$$

$$= \begin{bmatrix} \phi^2 p_{1,1} + q \\ \phi p_{1,1} \\ \phi p_{1,2} \\ \vdots \\ \phi p_{1,\ell-1} \\ \phi p_{1,\ell} \end{bmatrix} \tag{6.45}$$

and the innovations variance (a scalar)

$$H_{[s]}P_{[s],\,k}(-)H_{[s]}^{\mathrm{T}} + R_{[s]} \tag{6.46}$$

$$= \phi^2 p_{1,1} + q + r \tag{6.47}$$

so that the matrix product

$$P_{[s],\,k(-)}H_{[s]}^{\mathrm{T}}[H_{[s]}P_{[s],\,k(-)}H_{[s]}^{\mathrm{T}} + R]^{-1}H_{[s]}P_{[s],\,k(-)} = \frac{1}{p_{1,1}\phi^2+q+r}$$

$$\begin{bmatrix} \left(p_{1,1}\phi^2 + q\right)^2 & (p_{1,1}\phi^2 + q)p_{1,1}\phi & (p_{1,1}\phi^2 + q)p_{1,2}\phi & \cdots & (p_{1,1}\phi^2 + q)p_{1,\ell}\phi \\ (p_{1,1}\phi^2 + q)p_{1,1}\phi & {p_{1,1}}^2\phi^2 & p_{1,1}\phi^2 p_{1,2} & \cdots & p_{1,1}\phi^2 p_{1,\ell} \\ (p_{1,1}\phi^2 + q)p_{1,2}\phi & p_{1,1}\phi^2 p_{1,2} & {p_{1,2}}^2\phi^2 & \cdots & p_{1,2}\phi^2 p_{1,\ell} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ (p_{1,1}\phi^2 + q)p_{1,\ell}\phi & p_{1,1}\phi^2 p_{1,\ell} & p_{1,2}\phi^2 p_{1,\ell} & \cdots & {p_{1,\ell}}^2\phi^2 \end{bmatrix} \tag{6.48}$$

and the a posteriori smoother covariance

$$P_{[s],\,k(+)} = P_{[s],\,k(-)} - P_{[s],\,k(-)}H_{[s]}^{\mathrm{T}}[H_{[s]}P_{[s],\,k(-)}H_{[s]}^{\mathrm{T}} + R]^{-1}H_{[s]}P_{[s],\,k(-)} \tag{6.49}$$

$$= \begin{bmatrix} \phi^2 p_{1,1} + q & \phi p_{1,1} & \phi p_{1,2} & \cdots & \phi p_{1,\ell} \\ \phi p_{1,1} & p_{1,1} & p_{1,2} & \cdots & p_{1,\ell} \\ \phi p_{1,2} & p_{1,2} & p_{2,2} & \cdots & p_{2,\ell} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \phi p_{1,\ell} & p_{1,\ell} & p_{2,\ell} & \cdots & p_{\ell,\ell} \end{bmatrix} - \frac{1}{p_{1,1}\phi^2 + q + r}$$

$$\cdot \begin{bmatrix} \left(p_{1,1}\phi^2+q\right)^2 & (p_{1,1}\phi^2+q)p_{1,1}\phi & (p_{1,1}\phi^2+q)p_{1,2}\phi & \cdots & (p_{1,1}\phi^2+q)p_{1,\ell}\phi \\ (p_{1,1}\phi^2+q)p_{1,1}\phi & {p_{1,1}}^2\phi^2 & p_{1,1}\phi^2 p_{1,2} & \cdots & p_{1,1}\phi^2 p_{1,\ell} \\ (p_{1,1}\phi^2+q)p_{1,2}\phi & p_{1,1}\phi^2 p_{1,2} & {p_{1,2}}^2\phi^2 & \cdots & p_{1,2}\phi^2 p_{1,\ell} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ (p_{1,1}\phi^2+q)p_{1,\ell}\phi & p_{1,1}\phi^2 p_{1,\ell} & p_{1,2}\phi^2 p_{1,\ell} & \cdots & {p_{1,\ell}}^2\phi^2 \end{bmatrix} \cdot \tag{6.50}$$

At steady state, the a posteriori smoother covariance matrix of Equation 6.50 will equal the a posteriori smoother covariance matrix in Equation 6.39. By equating the matrix elements term by term, one gets the $\ell\ (\ell+1)/2$ equations:

$$p_{1,1} = p_{1,1}\phi^2 + q - \frac{(p_{1,1}\phi^2+q)^2}{p_{1,1}\phi^2+q+r} \tag{6.51}$$

$$p_{1,i+1} = p_{1,i}\phi - \frac{(p_{1,1}\phi^2+q)p_{1,i}\phi}{p_{1,1}\phi^2+q+r} \tag{6.52}$$

$$p_{i+1,i+1} = p_{i,i} - \frac{{p_{1,i}}^2\phi^2}{p_{1,1}\phi^2+q+r} \tag{6.53}$$

$$p_{j+1,j+k+1} = p_{j,j+k} - \frac{p_{1,j}\phi^2 p_{1,j+k}}{p_{1,1}\phi^2+q+r}, \tag{6.54}$$

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,

$$p_{1,1} = \frac{\phi^2 r - q - r + \sqrt{r^2(1-\phi^2)^2 + 2\ rq(\phi^2+1) + q^2}}{\phi^2}, \tag{6.55}$$

then with the cross-covariances of filter error with the smoother estimates,

$$p_{1,i+1} = \frac{p_{1,i}\phi\ r}{p_{1,1}\phi^2+q+r}, \tag{6.56}$$

for $1 \le i \le \ell$. Next, the variances of smoother stage estimates,

$$p_{i+1,i+1} = \frac{(p_{i,i}p_{1,1} - p_{1,i}^2)\phi^2 + p_{i,i}(q+r)}{p_{1,1}\phi^2+q+r}, \tag{6.57}$$

and finally the covariances between different smoother stage errors,

$$p_{j+1,j+k+1} = \frac{(p_{j,j+k}p_{1,1} - p_{1,j}p_{1,j+k})\phi^2 + p_{j,j+k}(q+r)}{p_{1,1}\phi^2+q+r} \tag{6.58}$$

for $1 \le j \le \ell - 1$ and $1 \le k \le \ell - j$.

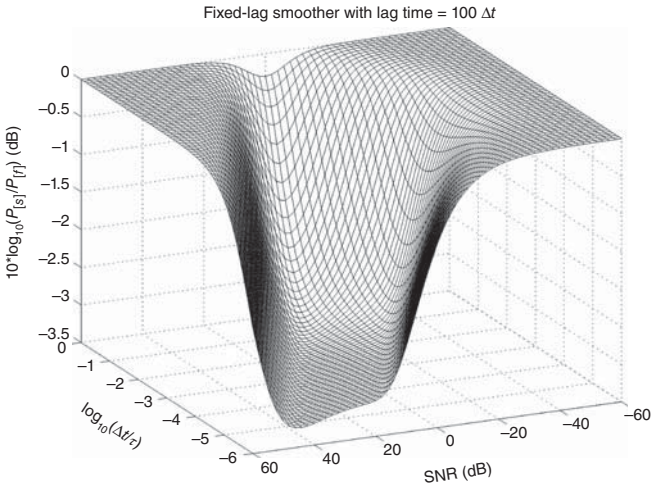


Figure 6.12 Improvement of smoothing over filtering versus SNR and $\Delta t/\tau$.

The ratio of ℓ-stage smoother variance to filter variance is then

$$\frac{\sigma^2_{[s]}}{\sigma^2_{[f]}} = \frac{p_{\ell+1,\ell+1}}{p_{1,1}}. \tag{6.59}$$

If the scalar $|\phi| < 1$, the dynamic system model is stable and models an exponentially correlated random process with finite steady-state mean-squared signal. Then the Riccati equation solution can be computed as a function of SNR and the ratio of discrete time step to correlation time. An example plot is shown in Figure 6.12, 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 two (−3 dB), which is as good as it gets for stable systems.

6.4 FIXED-POINT SMOOTHING

6.4.1 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 approach 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.

6.4.1.1 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 6.13. This shows the *forward Kalman–Bucy filter,* the *Kalman–Bucy predictor* from times before the designated fixed point in time t_{fixed}, and the *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 5.8.3 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.

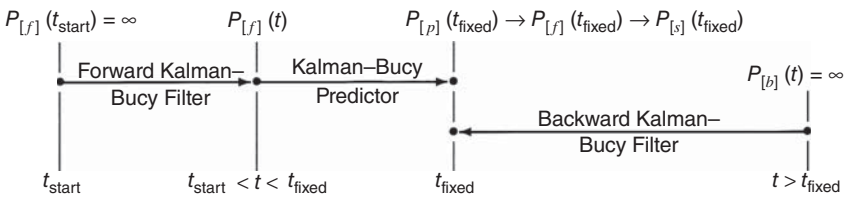


Figure 6.13 Kalman–Bucy fixed-point smoother model.

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 below in subsections 6.4.1.2 through 6.4.1.5.

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 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.

6.4.1.2 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

$$P_{[f]}(t) = A_{[f]}(t)/B_{[f]}(t)$$

$$\begin{bmatrix} A_{[f]}(t) \\ B_{[f]}(t) \end{bmatrix} = \exp\left(\begin{bmatrix} F & Q \\ H^2/R & -F \end{bmatrix} (t - t_{\text{start}}) \right) \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$P_{[f]}(t) = \frac{E_{[f]}^{(+)}\,\rho_{[f]} - RFE_{[f]}^{(-)} + RFE_{[f]}^{(+)} + E_{[f]}^{(-)}\,\rho_{[f]}}{H^2(-E_{[f]}^{(-)} + E_{[f]}^{(+)})}$$

$$\rho_{[f]} = \sqrt{R(F^2R + QH^2)}$$

$$E_{[f]}^{(-)} = e^{-\rho_{[f]}\,(t - t_{\text{start}})/R}$$

$$E_{[f]}^{(+)} = e^{\rho_{[f]}\,(t - t_{\text{start}})/R}.$$

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 $t_{\text{start}} < t \leq t_{\text{fixed}}$.

6.4.1.3 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

$$P_{[p]}(t_{\text{fixed}}) = \frac{A_{[p]}(t_{\text{fixed}})}{B_{[p]}(t_{\text{fixed}})}$$

$$\begin{bmatrix} A_{[p]}(t_{\text{fixed}}) \\ B_{[p]}(t_{\text{fixed}}) \end{bmatrix} = \exp\left(\begin{bmatrix} F & Q \\ 0 & -F \end{bmatrix} (t_{\text{fixed}} - t) \right) \begin{bmatrix} P_{[f]}(t) \\ 1 \end{bmatrix}$$

$$\begin{aligned} P_{[p]}(t_{\text{fixed}}) &= e^{2\ (t_{\text{fixed}} - t)\, F} P_{[f]}(t) + \frac{Q(e^{2\ (t_{\text{fixed}} - t)\, F} - 1)}{2\ F} \\ &= P_{[f]}(t) + Q(t_{\text{fixed}} - t) \text{ if } F = 0 \\ &= P_{[f]}(t_{\text{fixed}}) \text{ if } t = t_{\text{fixed}}. \end{aligned}$$

6.4.1.4 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

$$P_{[b]}(t_{\text{fixed}}) = \frac{A_{[b]}(t_{\text{fixed}})}{B_{[b]}(t_{\text{fixed}})}$$

$$\begin{bmatrix} A_{[b]}(t_{\text{fixed}}) \\ B_{[b]}(t_{\text{fixed}}) \end{bmatrix} = \exp\left(\begin{bmatrix} -F & Q \\ H^2/R & F \end{bmatrix} (t - t_{\text{fixed}}) \right) \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$P_{[b]}(t_{\text{fixed}}) = \frac{E_{[b]}^{(+)}\ \rho_{[b]} + RFE_{[b]}^{(-)} - RFE_{[b]}^{(+)} + E_{[b]}^{(-)}\ \rho_{[b]}}{H^2(-E_{[b]}^{(-)} + E_{[b]}^{(+)})}$$

$$\rho_{[b]} = \sqrt{R(F^2R + QH^2)}$$

$$E_{[b]}^{(-)} = e^{-\rho_{[b]}\ (t - t_{\text{fixed}})/R}$$

$$E_{[b]}^{(+)} = e^{\rho_{[b]}\ (t - t_{\text{fixed}})/R}.$$

6.4.1.5 Fixed-Point Smoother Submodel The resulting fixed-point smoother model uses the forward filter, predictor, and backward filter models, depending on where t is relative to t_{fixed}:

$$P_{[s]}(t_{\text{fixed}}) = \begin{cases} P_{[p]}\left(t_{\text{fixed}}\right), & t < t_{\text{fixed}} \\ P_{[f]}(t_{\text{fixed}}), & t = t_{\text{fixed}} \\ \frac{P_{[f]}(t_{\text{fixed}})P_{[b]}(t_{\text{fixed}})}{P_{[f]}(t_{\text{fixed}})+P_{[b]}(t_{\text{fixed}})}, & t > t_{\text{fixed}} \end{cases},$$

where the subscript $[s]$ on $P_{[s]}$ indicates the variance of estimation uncertainty of the fixed-point smoother.

6.4.1.6 Fixed-Point Smoothing Performance Figure 6.14 is a plot of modeled RMS fixed-point smoother uncertainty for a stochastic process model with

$$F = 0.01 \text{ (dynamic coefficient)},$$
$$Q = 0.01 \text{ (variance of dynamic process noise)},$$
$$H = 1 \text{ (measurement sensitivity)},$$
$$R = 0.01 \text{ (variance of measurement noise)},$$
$$t_{\text{start}} = 0 \text{ (starting time)},$$
$$t_{\text{fixed}} = 100 \text{ (fixed time at which system state is to be estimated)},$$
$$t_{\text{end}} = 300 \text{ (ending time)},$$

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

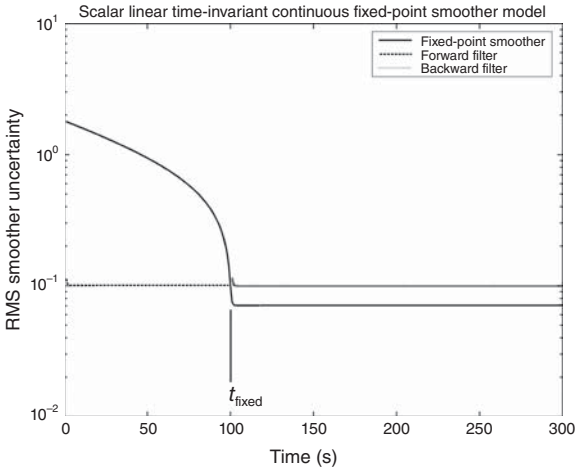


Figure 6.14 Sample performance of fixed-point smoother with $t_{\text{fixed}} = 100$.

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}.

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 6.15 (varying Q), 6.16 (varying R), and 6.17 (varying F), which are multiplots of the results with different parameter values.

6.4.2 Discrete-Time Fixed-Point Smoother

This type of smoother includes the 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 $t_i < t_k$:

$$\hat{x}_{[s]i|k} = \hat{x}_{[s]i|k-1} + B_k \overline{K}_k (z_k - H\hat{x}_{k(-)}), \tag{6.60}$$

$$B_k = B_{k-1} P_{k-1(+)} \Phi_{k-1}^{\mathrm{T}} P_{k(-)}^{-1}, \tag{6.61}$$

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 INS test data may be found in Reference 1.) The values of

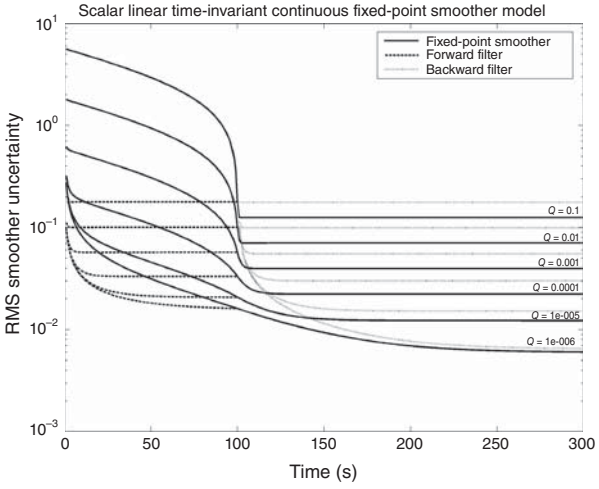


Figure 6.15 Fixed-point smoother performance with varying Q.

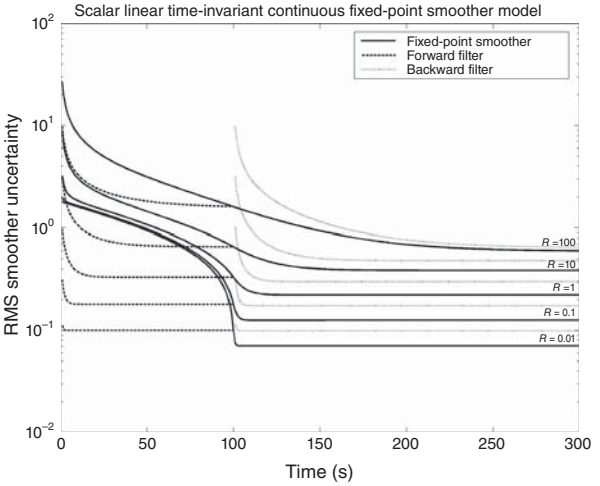


Figure 6.16 Fixed-point smoother performance with varying R.

$\hat{x}_{k(-)}, \overline{K}_k, z_k, H_k P$, 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

$$P_{[s]i|k} = P_{[s]i|k-1} + B_k(P_{k(+)} - P_{k(-)})B_k^{\mathrm{T}}, \tag{6.62}$$

although this is not a necessary part of the smoother implementation.

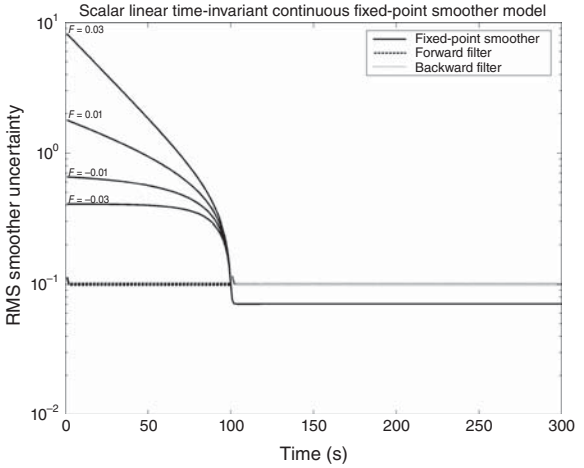


Figure 6.17 Fixed-point smoother performance with varying F.

6.5 SUMMARY

6.5.1 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.

6.5.1.1 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. It is most commonly used in post-processing.

fixed-lag smoothers, which typically run in real time but generate an estimated 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 $t - \Delta t_{\text{lag}}$. Some of the first fixed-lag smoother implementations were numerically unstable, a problem that 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}.

6.5.1.2 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.

6.5.2 Improvement of Smoothing over Filtering

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 sensor model (H and R).
- For stable dynamic system models, the improvement of smoothing over filtering is at most a factor of two.

- 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.

6.5.3 Sources for Additional Information

For historical and technical accounts of seminal works in optimal smoothing, see the surveys by Meditch [18], Kailath [19], and Park and Kailath [5]. 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 [20].

PROBLEMS

6.1 For a scalar system with $\Phi = 1$, $H = 1$, $R = 2$, and $Q = 1$, find the steady-state value of the covariance P for a fixed-point smoother estimate.

6.2 Find the solution to Problem 5.1 for P_{10} when $P_0 = 1$, using the fixed-interval smoother on the interval $0 \leq k \leq 10$.

6.3 Solve Problem 5.1 with a fixed-lag smoother with the lag equal to two time steps.

6.4 Repeat Problem 5.1 with $R = 15$.

6.5 Let the model parameters of the Kalman filter be

$$\Phi_{[f]} = \begin{bmatrix} 0.9 & 0.3 \\ -0.3 & 0.9 \end{bmatrix} \tag{6.63}$$

$$H_{[f]} = \begin{bmatrix} 1 & 0 \end{bmatrix} \tag{6.64}$$

$$Q_{[f]} = \begin{bmatrix} 0.25 & 0 \\ 0 & 0.25 \end{bmatrix} \tag{6.65}$$

$$R_{[f]} = 1. \tag{6.66}$$

For a compatible BMFLS implementation with a lag of $\ell = 3$ 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 $Q_{[s]}$.

6.6 For the 2×2 matrix $\Phi_{[f]}$ of Equation 6.63,

(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

$$x_k = \Phi[f]^k x_0 \text{ for } k = 1,\ 2,\ 3,\ \ldots,\ 100.$$

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 make more than a factor of two improvementa in mean-squared estimation uncertainty? Justify your answer.

6.7 Repeat the exercises of the previous problem with

$$\Phi_{[f]} = \begin{bmatrix} 0.9 & 0.9 \\ -0.9 & 0.9 \end{bmatrix}. \tag{6.67}$$

6.8 Use the parameter values of Problem 6.5 and the MATLAB function `BMFLS` on the companion Wiley web site 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

1. The simulated state vector.
2. The filter estimate.
3. The smoother estimate.
4. RMS filter uncertainties.
5. RMS smoother uncertainties.

Compare your results to those shown in Figure 6.11 of Example 6.1.

6.9 Use the parameter values of Problem 6.7 and the MATLAB function `BMFLS` on the companion Wiley web site to demonstrate fixed-lag smoothing on simulated measurements. Perform all the same tasks as in Problem 6.8.

REFERENCES

[1] 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, IEEE, New York, pp. 476–479, 1988.

[2] B. D. O. Anderson, "Properties of optimal linear smoothing," *IEEE Transactions on Automatic Control*, Vol. AC-14, pp. 114–115, 1969.

[3] B. D. O. Anderson and S. Chirarattananon, "Smoothing as an improvement on filtering: a universal bound," *Electronics Letters*, Vol. 7, No. 18, pp. 524–525, 1971.

[4] 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.

[5] P. G. Park and T. Kailath, "New square-root smoothing algorithms," *IEEE Transactions on Automatic Control*, Vol. 41, pp. 727–732, 1996.

[6] 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.

[7] H. E. Rauch, "Solutions to the linear smoothing problem," *IEEE Transactions on Automatic Control*, Vol. AC-8, pp. 371–372, 1963.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] J. B. Moore, "Discrete-time fixed-lag smoothing algorithms," *Automatica*, Vol. 9, No. 2, pp. 163–174, 1973.

[15] S. Prasad and A. K. Mahalanabis, "Finite lag receivers for analog communication," *IEEE Transactions on Communications*, Vol. 23, pp. 204–213, 1975.

[16] 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.

[17] J. F. Riccati, "Animadversationnes in aequationes differentiales secundi gradus," *Acta Eruditorum Quae Lipside Publicantur Supplementa*, Vol. 8, pp. 66–73, 1724.

[18] J. S. Meditch, "A survey of data smoothing for linear and nonlinear dynamic systems," *Automatica*, Vol. 9, pp. 151–162, 1973.

[19] T. Kailath, "Correspondence item," *Automatica*, Vol. 11, pp. 109–111, 1975.

[20] S. R. McReynolds, "Fixed interval smoothing: revisited," *AIAA Journal of Guidance, Control, and Dynamics*, Vol. 13, pp. 913–921, 1990.

7

IMPLEMENTATION METHODS

There is a great difference between theory and practice.

Giacomo Antonelli (1806–1876)[1]

7.1 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* [1], 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®, Fourth Edition.
Mohinder S. Grewal and Angus P. Andrews.
© 2015 John Wiley & Sons, Inc. Published 2015 by John Wiley & Sons, Inc.

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.

7.1.1 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 the so-called "square roots" of the covariance matrix are used as the dependent variables.
6. Numerical methods for solving the Riccati equation in terms of these matrix "square roots" 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.

7.1.2 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 in 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 [2] (methods up to

1973) and McReynolds [3] (up to 1990) and to earlier results by Bierman [4] and by Watanabe and Tzafestas [5].

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 [6].

7.2 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 7.1 (Roundoff Errors) In binary representation, the rational numbers are transformed into sums of powers of 2, as follows:

$$\begin{aligned} 1 &= 2^0 \\ 3 &= 2^0 + 2^1 \\ \frac{1}{3} &= \frac{1}{4} + \frac{1}{16} + \frac{1}{64} + \frac{1}{256} + \cdots \\ &= 0_{\text{b}}0101010101010101010101010 \ \ldots , \end{aligned}$$

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 IEEE/ANSI standard [7] 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"):

$$\begin{aligned} \frac{1}{3} &\approx 0_{\text{b}}010101010101010101010101011 \\ &= \frac{11184811}{33554432} \\ &= \frac{1}{3} - \frac{1}{100663296}, \end{aligned}$$

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*.

[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.

7.2.1 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

$$1 + \varepsilon_{\text{roundoff}} \equiv 1 \text{ in machine precision} \tag{7.1}$$

or

$$1 + \varepsilon_{\text{roundoff}}/2 \equiv 1 \text{ in machine precision.} \tag{7.2}$$

The name "`eps`" in MATLAB®is the parameter satisfying the second of these equations. Its value may be found by typing "`eps`⟨↩⟩" (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.

7.2.2 Effects of Roundoff on Kalman Filter Performance

7.2.2.1 Early Discoveries Around the time the Kalman filter was introduced in 1960, the International Business Machines Corporation (IBM) was introducing its 7000-series transisterized 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 Schmidt [8, 9] and Battin [10] 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 wordlengths than those available for modern MATLAB implementations and with less accurate implementations of bit-level arithmetic than current microprocessors adhering to current ANSI standards [7].

However, the next example (modified after one from Dyer and McReynolds [11] credited to R. 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 7.2 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_0H^{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 $HP_0H^{\mathrm{T}} + R$ is not invertible.

Sneak Preview of Alternative Implementations Figure 7.1 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 MATLAB m-file `shootout.m` on the accompanying diskette) 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 diskette. 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 $\delta \to \varepsilon$, the machine precision limit. The Carlson and Bierman solutions still maintain about nine 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.

7.2.3 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.

7.2.3.1 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.

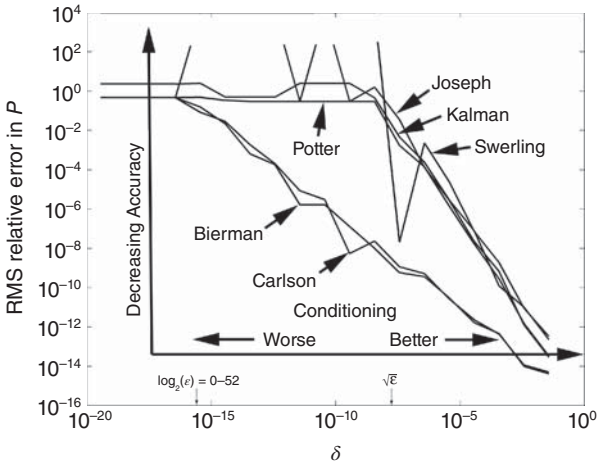


Figure 7.1 Degradation of the Riccati equation observational updates with problem conditioning.

7.2.3.2 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.

7.2.3.3 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.

7.2.3.4 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 7.3 (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

$$\text{cond}(A) = \frac{\max_x \|Ax\|/\|x\|}{\min_x \|Ax\|/\|x\|} \tag{7.3}$$

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 $c_A \epsilon_{\text{roundoff}} \text{cond}(A)$, where $\epsilon_{\text{roundoff}}$ is the unit roundoff error in computer arithmetic (defined in Section 7.2.1) 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 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.

7.2.4 Ill-Conditioned Kalman Filtering Problems

For Kalman filtering problems, the solution of the associated Riccati equation should equal the covariance matrix of 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.

7.3 EFFECTS OF ROUNDOFF ERRORS ON KALMAN FILTERS

7.3.1 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 [12] 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.

7.3.2 Roundoff Error Propagation in Kalman Filters

7.3.2.1 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 that is shown in Figure 7.2. This figure shows the data flow at the level of vectors 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.)

7.3.2.2 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 by the feedback mechanism, so long as the loop gain is correct. That gain is computed in the other loop.

7.3.2.3 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 [13] that the gain loop, by itself,

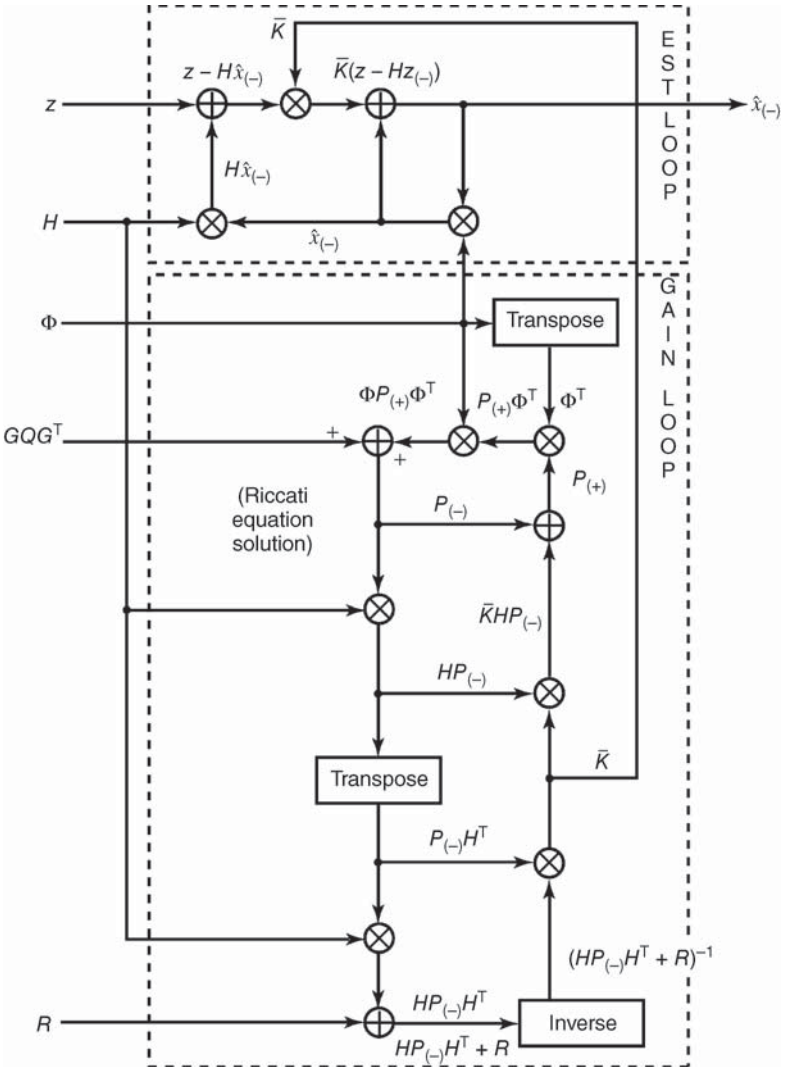


Figure 7.2 Kalman filter data flow.

is not unstable. However, even bounded errors in the computed value of P may momentarily destabilize the estimation loop.

Example 7.4 An illustration of the effects that negative characteristic values of the computed covariance matrix P can have on the estimation errors is shown as follows.

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.

7.3.2.4 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

$$\delta x_{k+1(-)} = f_1(\delta x_{k(-)}, \delta P_{k(-)}) + \Delta x_{k+1}, \tag{7.4}$$

$$\delta P_{k+1(-)} = f_2(\delta P_{k(-)}) + \Delta P_{k+1(-)}, \tag{7.5}$$

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 7.1. 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 [12]. (Many of these results have also appeared in earlier publications.) These expressions represent the first-order error in the updated a prior 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.

TABLE 7.1 First-Order Error Propagation Models

Roundoff Error in Filter Variable	Error Model (by Filter Type)	
	Conventional Implementation	Square-Root Covariance
$\delta x_{k+1(-)}$ $\delta \overline{K}_k$	$A_1[\delta x_{k(-)} + \delta P_{k(-)} A_2(z - H x_{k(-)})] + \Delta x_{k+1}$ $A_1 \delta P_{k(-)}$	
$\delta P_{k+1(-)}$	$A_1 \delta P_{k(-)} A_1^{\mathrm{T}} + \Delta P_{k+1}$ $+\Phi(\delta P_{k(-)} - \delta P_{k(-)}^{\mathrm{T}})\Phi^{\mathrm{T}}$ $-\Phi(\delta P_{k(-)} - \delta P_{k(-)}^{\mathrm{T}})A_1^{\mathrm{T}}$	$A_1 \delta P_{k(-)} A_1^{\mathrm{T}}$ $+\Delta P_{k+1}$

Notes: $A_1 = \Phi - \overline{K}_k H$; $\quad A_2 = H^{\mathrm{T}}[H P_k H^{\mathrm{T}} + R]^{-1}$.

Roundoff Error Propagation Table 7.1 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 5. 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.

Propagation of Antisymmetry Errors Note the two terms in Table 7.1 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 $\Phi P \Phi^{\mathrm{T}}, HPH^{\mathrm{T}}, HPH^{\mathrm{T}} + R$, 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 7.2. The tabulated formulas are upper bounds on these random errors.

TABLE 7.2 Upper Bounds on Added Roundoff Errors

	Upper Bounds (by Filter Type)	
Norm of Roundoff Errors	Conventional Implementation	Square-Root Covariance
$\lvert\Delta x_{k+1(-)}\rvert$	$\varepsilon_1(\lvert A_1\rVert x_{k(-)}\rvert + \lvert\overline{K}_k\rVert z_k\rvert)$ $+ \lvert\Delta\overline{K}_k\rvert(\lvert H\rVert x_{k(-)}\rvert + \lvert z_k\rvert)$	$\varepsilon_4(\lvert A_1\rVert x_{k(-)}\rvert + \lvert\overline{K}_k\rVert z_k\rvert)$ $+ \lvert\Delta\overline{K}_k\rvert(\lvert H\rVert x_{k(-)}\rvert + \lvert z_k\rvert)$
$\lvert\Delta\overline{K}_k\rvert$	$\varepsilon_2\kappa^2(R^*)\lvert\overline{K}_k\rvert$	$\varepsilon_5\kappa(R^*)[\lambda_m^{-1}(R^*)\lvert C_{p(k+1)}\rvert$ $+\lvert\overline{K}_k C_{R^*}\rvert + \lvert A_3\rvert/\lambda_1(R^*)]$
$\lvert\Delta P_{k+1(-)}\rvert$	$\varepsilon_3\kappa^2(R^*)\lvert P_{k+1(-)}\rvert$	$\dfrac{\varepsilon_6[1+\kappa(R^*)]\lvert P_{k+1}\rVert A_3\rvert}{\lvert C_{p(k+1)}\rvert}$

Notes: $\varepsilon_1, \ldots, \varepsilon_6$ are constant multiples of ε, the unit roundoff error; $A_1 = \Phi - \overline{K}_k H; A_3 = [(\overline{K}_k C_{R^*})\lvert C_{p(k+1)}]; R^* = HP_{k(-)}H^{\mathrm{T}} + R; R^* = C_{R^*}C_{R^*}^{\mathrm{T}}$ (triangular Cholesky decomposition); $P_{k+1(-)} = C_{p(k+1)}C_{p(k+1)}^{\mathrm{T}}$ (triangular Cholesky decomposition); $\lambda_1(R^*) \geq \lambda_2(R^*) \geq \cdots \geq \lambda_\ell(R^*) \geq 0$ are the characteristic values of R^*; $\kappa(R^*) = \lambda_1(R^*)/\lambda_\ell(R^*)$ is the condition number of R^*.

The important points which these tables demonstrate 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 into 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 have obtained upper bounds on the norms of the added errors Δx, $\Delta\overline{K}$, and ΔP, as shown in Table 7.2. 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 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 E.) 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 [12] 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 [14].

7.3.3 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 7.5 (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 so that, in the 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 table below:

Observation Number	Expression	Value: Exact	Value: Rounded
1	$P_0 H^{\mathrm{T}}$	P_0	P_0
1	$HP_0 H^{\mathrm{T}}$	P_0	P_0
1	$HP_0 H^{\mathrm{T}} + R$	$P_0 + R$	P_0
1	$\overline{K}_1 = P_0 H^{\mathrm{T}} (HP_0 H^{\mathrm{T}} + R)^{-1}$	$\dfrac{P_0}{P_0 + R}$	1
1	$P_1 = P_0 - \overline{K}_1 HP_0$	$\dfrac{P_0 R}{P_0 + R}$	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
	$P_k = P_{k-1} - \overline{K}_k HP_{k-1}$	$\dfrac{P_0 R}{kP_0 + R}$	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 $P_0 R/(P_0 + R) \approx R$ (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 $P_0 R/(10P_0 + R) \approx \frac{1}{10} R$.

The ill-conditioning in this example is due to the misscaling between the a priori state estimation uncertainty and the measurement uncertainty.

7.4 FACTORIZATION METHODS FOR "SQUARE-ROOT" FILTERING

7.4.1 Background

The historical background and notational ambiguity of "square-root" filtering were discussed in Chapter 1.

7.4.2 Types of Cholesky Factors

We need to distinguish three different types of Cholesky factors for symmetric (and therefore square), positive-definite matrices:

1. *Cholesky factors* are triangular (and therefore square) with positive diagonal entries. They may be either
   (a) lower triangular (i.e., with zeros above the diagonal), or
   (b) upper triangular (i.e., with zeros below the diagonal).
2. *Generalized Cholesky factors* C of a matrix P are not required to be triangular—or even square—but their symmetric product must satisfy the equation $CC^{\mathrm{T}} = P$. All Cholesky factors are also generalized Cholesky factors, but not all generalized Cholesky factors are Cholesky factors. Some of the methods used in "square-root" filtering are for converting generalized Cholesky factors into true Cholesky factors.
3. *Modified Cholesky factorization* has the form

$$P = UDU^{\mathrm{T}},$$

   where U is a *unit triangular matrix* (i.e., with ones on its diagonal) and D is a diagonal matrix with positive diagonal entries. Modified Cholesky factors are neither ordinary Cholesky factors nor generalized Cholesky factors.

Square-root filtering reformulates the Riccati equation to replace the covariance matrix with its "square root" (actually, a Cholesky factor—either straight, generalized, or modified). Either way, this makes a significant difference in the numerical stability of the solution, even though it becomes necessary to "square" the result (actually, using the symmetric product) to recover the covariance matrix for analysis.

However, there are many ways to implement "square root" formulations of the Kalman filter, depending on which type of factoring is used. There are also many ways such matrix factoring methods can be used in different parts of the Kalman filter.

7.4.3 Overview of Matrix Factorization Tricks

See Chapter 1 for distinctions between matrix decompositions, factorizations, and triangularizations. A further distinction between *decomposition* and *factorization* is made by Dongarra et al. [15], 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. These include *triangularization,* used to denote *QR* factorizations (in the sense of Dongarra et al.) involving a triangular factor that is preserved and an orthogonal factor that is not preserved.

7.4.3.1 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 (triangular

with positive diagonal elements), generalized Cholesky factors, or 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 transformed 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 and R already have special meanings in Kalman filtering. The two triangularization methods used in Kalman filtering are
      i. *Givens rotations* [164] triangularize a matrix by operating on one element at a time. (A *modified Givens method* due to Gentleman [163] generates *diagonal* and *unit triangular* factors.)
      ii. *Householder transformations* [21] 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 generalized Cholesky factors C and $[A\ B]$ must then be related by orthogonal transformations (triangularizations). A QR decomposition of

[A 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 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 generalized Cholesky factors.
2. The *observational update* equation is solved in "square-root" form by using alternative 2×2 block-partitioned generalized 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 generalized 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 7.5.2–7.7.2 and 7.7. They make use of the general numerical procedures presented in Sections 7.4.4–7.4.7.

7.4.4 Cholesky Decomposition Methods and Applications

7.4.4.1 Symmetric Products and Generalized 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 the *generalized Cholesky factor* of M (Section B.6 of Appendix B on the Wiley website). Strictly speaking, a generalized 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 generalized Cholesky factors, but the generalized 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 generalized Cholesky factor of M. Transformations of one generalized 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 7.4.4.3);
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 generalized Cholesky factors are used for generating independent random sequences of vectors with pre-specified means and covariance matrices (see Section 4.3.9).

7.4.4.2 Decomposition Algorithms

Symmetric Square Roots The *singular-value decomposition* of a symmetric positive-definite matrix P has the form

$$P = \mathcal{E}\Lambda\mathcal{E}^{\mathrm{T}} \tag{7.6}$$

$\mathcal{E}$ = 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

$$P = \sum_{j=1}^{n} \lambda_j \, e_j e_j^{\mathrm{T}}. \tag{7.7}$$

They also define a symmetric generalized Cholesky factor of P as

$$C \stackrel{\text{def}}{=} \sum_{j=1}^{n} \sqrt{\lambda_j} \, e_j e_j^{\mathrm{T}}, \tag{7.8}$$

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

$$CC^{\mathrm{T}} = \left[\sum_{j=1}^{n} \sqrt{\lambda_j} \, e_j e_j^{\mathrm{T}}\right] \left[\sum_{k=1}^{n} \sqrt{\lambda_k} \, e_k e_k^{\mathrm{T}}\right]^{\mathrm{T}} \tag{7.9}$$

$$= \sum_{j=1}^{n} \sum_{k=1}^{n} \sqrt{\lambda_j}\ \sqrt{\lambda_k}\ e_j \{e_j^{\mathrm{T}}\ e_k\} e_k^{\mathrm{T}} \tag{7.10}$$

$$\{e_j^{\mathrm{T}}\ e_k\} = \begin{cases} 0, & j \neq k \\ 1, & j = k \end{cases} \tag{7.11}$$

$$CC^{\mathrm{T}} = \sum_{j=1}^{n} \left[\sqrt{\lambda_j}\right]^2 e_j e_j^{\mathrm{T}} \tag{7.12}$$

$$CC^{\mathrm{T}} = \sum_{j=1}^{n} \lambda_j\ e_j e_j^{\mathrm{T}} \tag{7.13}$$

$$= P. \tag{7.14}$$

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 generalized 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 matrix square roots are not that popular in Kalman filtering, because there are much simpler algorithms for computing triangular Cholesky factors, but symmetric generalized Cholesky factors may be useful in sample-based methods for nonlinear filtering.

Triangular Cholesky Factors Recall that the *main diagonal* of an $n \times m$ matrix C is the set of elements $\{C_{ii} | 1 \leq i \leq \min(m, n)\}$ 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 the 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 7.6 **(3×3 example)** Consider the 3×3 example for finding a lower triangular Cholesky factor $P = CC^{\mathrm{T}}$ for symmetric P:

$$\begin{bmatrix} p_{11} & p_{21} & p_{31} \\ p_{21} & p_{22} & p_{32} \\ p_{31} & p_{32} & p_{33} \end{bmatrix} = \begin{bmatrix} c_{11} & 0 & 0 \\ c_{21} & c_{22} & 0 \\ c_{31} & c_{32} & c_{33} \end{bmatrix} \begin{bmatrix} c_{11} & 0 & 0 \\ c_{21} & c_{22} & 0 \\ c_{31} & c_{32} & c_{33} \end{bmatrix}^{\mathrm{T}}$$

$$= \begin{bmatrix} c_{11}^2 & c_{11}c_{21} & c_{11}c_{31} \\ c_{11}c_{21} & c_{21}^2 + c_{22}^2 & c_{21}c_{31} + c_{22}c_{32} \\ c_{11}c_{31} & c_{21}c_{31} + c_{22}c_{32} & c_{31}^2 + c_{32}^2 + c_{33}^2 \end{bmatrix}.$$

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 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 7.3. 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 `utchol.m`.

Programming Note: MATLAB automatically assigns the value zero to all the unassigned matrix locations. This would 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.

7.4.4.3 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.

TABLE 7.3 Cholesky Decomposition Algorithms

Given an $m \times m$ symmetric positive-definite matrix M, computes a triangular matrix C such that $M = CC^{\mathrm{T}}$.

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 + $m\sqrt{\ }$.

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 7.4. This algorithm is implemented in the m-file `modchol.m`. It takes M as input and 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 7.3. The big difference is that the modified Cholesky decomposition does not require taking scalar square roots.

7.4.4.4 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[3] of the Kalman filter from this approach will be covered in Section 7.7.1.

[3]The methodology used for determining the computational complexities of algorithms in this chapter is presented in Section 7.4.4.7.

TABLE 7.4 *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.

Suppose, for example, that

$$z = Hx + \xi \tag{7.15}$$

is an observation with measurement sensitivity matrix H and noise ξ that is correlated from component to component of ξ. That is, the covariance matrix

$$\mathrm{E}\langle \xi \xi^{\mathrm{T}} \rangle = R \tag{7.16}$$

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

$$R = UDU^{\mathrm{T}}, \tag{7.17}$$

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

$$\acute{z} = U^{-1}z \tag{7.18}$$
$$= U^{-1}(Hx + \xi) \tag{7.19}$$
$$= (U^{-1}H)x + (U^{-1}\xi) \tag{7.20}$$
$$= \acute{H}x + \acute{\xi}. \tag{7.21}$$

That is, this "new" measurement $\acute{z}$ has measurement sensitivity matrix $\acute{H} = U^{-1}H$ and observation error $\acute{\xi} = U^{-1}\xi$. The covariance matrix R' of the observation error $\acute{\xi}$ will be the expected value

$$R' = \mathrm{E}\langle\acute{\xi}\acute{\xi}^{\mathrm{T}}\rangle \tag{7.22}$$
$$= \mathrm{E}\langle(U^{-1}\xi)(U^{-1}\xi)^{\mathrm{T}}\rangle \tag{7.23}$$
$$= \mathrm{E}\langle U^{-1}\xi\xi^{\mathrm{T}}U^{\mathrm{T}-1}\rangle \tag{7.24}$$
$$= U^{-1}\mathrm{E}\langle\xi\xi^{\mathrm{T}}\rangle U^{\mathrm{T}-1} \tag{7.25}$$
$$= U^{-1}RU^{\mathrm{T}-1} \tag{7.26}$$
$$= U^{-1}(UDU^{\mathrm{T}})U^{\mathrm{T}-1} \tag{7.27}$$
$$= D. \tag{7.28}$$

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

$$U\acute{z} = z \tag{7.29}$$
$$U\acute{H} = H \tag{7.30}$$

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 7.5 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.

TABLE 7.5 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: $Y := U^{-1}Y$ (In-Place)

`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.	

Specialization for Measurement Decorrelation A complete procedure for measurement decorrelation is listed in Table 7.6. 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}).

7.4.4.5 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

$$[UDU^{\mathrm{T}}][HPH^{\mathrm{T}} + R]^{-1}H = H. \tag{7.31}$$

It then suffices to solve

$$UDU^{\mathrm{T}}X = H \tag{7.32}$$

for X. This can be done by solving the three problems

$$UX_{[1]} = H \text{ for } X_{[1]}, \tag{7.33}$$

$$DX_{[2]} = X_{[1]} \text{ for } X_{[2]}, \tag{7.34}$$

$$U^{\mathrm{T}}X = X_{[2]} \text{ for } X. \tag{7.35}$$

TABLE 7.6 Measurement Decorrelation Procedure

The vector-valued measurement $z = Hx + v$, with correlated components of the measurement error $E(vv^{\mathrm{T}}) = R$, is transformed to the measurement $\acute{z} = \acute{H}x + \acute{v}$ with uncorrelated components of the measurement error $\acute{v} : E\langle\acute{v}\acute{v}^{\mathrm{T}}\rangle = \mathrm{D}$, 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.
2. Solve $U\acute{z} = z$ and $U\acute{H} = H$ in place.

Computational complexity: $\frac{1}{6}\ell(\ell-1)(\ell+4) + \frac{1}{2}\ell(\ell-1)(n+1)$ flops.

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.

7.4.4.6 Transforming Covariance Matrices to 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 just 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 7.4. A procedure for inverting the result in place is shown in Table 7.7. A matrix inversion procedure using these two is outlined in Table 7.8. It should be used with caution, however.

7.4.4.7 Computational Complexities Using the general methods outlined in References 16 and 17, one can derive the complexity formulas shown in Table 7.9 for methods using generalized Cholesky factors.

TABLE 7.7 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.

7.4.5 Kalman Implementation with Decorrelation

It was pointed out by Kaminski [18] 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 7.6, 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 7.10. 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.

7.4.6 Symmetric Square Roots of Elementary Matrices

7.4.6.1 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 **v, 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 (**v, w**) but with different scalar values (s).

TABLE 7.8 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: $(m-1)(2m+5)/6$ flops

7.4.6.2 Symmetric Elementary Matrices An elementary matrix is symmetric if $v = w$. The squares of such matrices have the same format:

$$(I - \sigma \mathbf{v}\mathbf{v}^{\mathrm{T}})^2 = (I - \sigma \mathbf{v}\mathbf{v}^{\mathrm{T}})(I - \sigma \mathbf{v}\mathbf{v}^{\mathrm{T}}) \tag{7.36}$$

$$= I - 2\sigma \mathbf{v}\mathbf{v}^{\mathrm{T}} + \sigma^2 |\mathbf{v}|^2 \mathbf{v}\mathbf{v}^{\mathrm{T}} \tag{7.37}$$

$$= I - (2\sigma - \sigma^2 |\mathbf{v}|^2)\mathbf{v}\mathbf{v}^{\mathrm{T}} \tag{7.38}$$

$$= I - s\mathbf{v}\mathbf{v}^{\mathrm{T}} \tag{7.39}$$

$$s = (2\sigma - \sigma^2 |\mathbf{v}|^2). \tag{7.40}$$

7.4.6.3 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

$$s = 2\sigma - \sigma^2 |\mathbf{v}|^2, \tag{7.41}$$

$$\sigma^2 |\mathbf{v}|^2 - 2\sigma + s = 0 \tag{7.42}$$

TABLE 7.9 Computational Complexity Formulas

Cholesky decomposition of an $m \times m$ matrix:

$$C_{\text{Cholesky}} = \sum_{j=1}^{m} \left[m - j + \sum_{i=j}^{m} (m - j) \right]$$
$$= \frac{1}{3}m^3 + \frac{1}{2}m^2 - \frac{5}{6}m$$

UD decomposition of an $m \times m$ matrix:

$$C_{UD} = \sum_{j=1}^{m} \left[m - j + \sum_{i=j}^{m} 2(m - j) \right]$$
$$= \frac{2}{3}m^3 + \frac{1}{2}m^2 - \frac{7}{6}m$$

Inversion of an $m \times m$ unit triangular matrix:

$$C_{\text{UTINV}} = \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} (j - i - 1)$$
$$= \frac{1}{6}m^3 - \frac{1}{2}m^2 + \frac{1}{3}m$$

Measurement decorrelation ($\ell \times n$ H-matrix):

$$C_{\text{DeCorr}} = C_{UD} + \sum_{i=1}^{\ell-1} \sum_{k=i+1}^{\ell} (n + 1)$$
$$= \frac{2}{3}\ell^3 + \ell^2 - \frac{5}{3}\ell + \frac{1}{2}\ell^2 n - \frac{1}{2}\ell n$$

Inversion of an $m \times m$ covariance matrix:

$$C_{\text{COVINV}} = C_{UD} + C_{\text{UTINV}} + m + \sum_{i=1}^{m} [i(i - 1)(m - i + 1)]$$
$$= m^3 + \frac{1}{2}m^2 + \frac{1}{2}m$$

TABLE 7.10 Operations for Sequential Processing of Measurements

Operation	Flops
$H \times P_{(-)}$	n^2
$H \times [HP_{(-)}]^{\mathrm{T}} + R$	n
$\{H[HP_{(-)}]^{T} + R\}^{-1}$	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) $\times \ell$ components	$(\frac{1}{2}n^2 + \frac{5}{2}n + 1) \times \ell$
+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	$\frac{2}{3}\ell^3 + \ell^2 - \frac{2}{3}\ell + \frac{1}{2}\ell^2 n + 2\ell + n\frac{1}{2}\ell n^2$

to obtain the solution

$$(I - s\mathbf{v}\mathbf{v}^{\mathrm{T}})^{1/2} = I - \sigma\mathbf{v}\mathbf{v}^{\mathrm{T}}, \tag{7.43}$$

$$\sigma = \frac{1 + \sqrt{1 - s|\mathbf{v}|^2}}{|\mathbf{v}|^2}. \tag{7.44}$$

In order that this square root be a real matrix, it is necessary that the radicand

$$1 - s|\mathbf{v}|^2 \geq 0. \tag{7.45}$$

7.4.7 Triangularization Methods

7.4.7.1 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.

7.4.7.2 Triangularization (QR Decomposition) of A It is a theorem of linear algebra that any general matrix A can be represented as a product[4]

$$A = C_{k+1(-)}\mathrm{T} \tag{7.46}$$

[4]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

of a triangular matrix $C_{k+1(-)}$ and an orthogonal matrix T. This type of decomposition is called *QR decomposition* or *triangularization*. By means of this triangularization, the symmetric matrix product factorization

$$P_{k+1(-)} = AA^{\mathrm{T}} \tag{7.47}$$

$$= [C_{k+1(-)}T][C_{k+1(-)}T]^{\mathrm{T}} \tag{7.48}$$

$$= C_{k+1(-)}TT^{\mathrm{T}}C_{k+1(-)}^{\mathrm{T}} \tag{7.49}$$

$$= C_{k+1(-)}(TT^{\mathrm{T}})C_{k+1(-)}^{\mathrm{T}} \tag{7.50}$$

$$= C_{k+1(-)}C_{k+1(-)}^{\mathrm{T}} \tag{7.51}$$

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.

7.4.7.3 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 generalized Cholesky factors of the covariance matrix of estimation uncertainty, as described above;
- observational updates of generalized Cholesky factors of the estimation information matrix, as described in Section 7.7.3.5; and
- combined updates (observational and temporal) of generalized Cholesky factors of the covariance matrix of estimation uncertainty, as described in Section 7.7.2.

A modified Givens rotation due to Gentleman [19] 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.

7.4.7.4 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:

$$T = T_1T_2T_3 \cdots T_m. \tag{7.52}$$

committed the symbols Q and R to play other roles in this book. In this instance of the QR decomposition, it has the transposed form $A^{\mathrm{T}} = T^{\mathrm{T}}C_{k+1(-)}^{\mathrm{T}}$, where T^{T} is the stand-in for the original Q (the orthogonal factor) and $C_{k+1(-)}^{\mathrm{T}}$ is the stand-in for the original R (the triangular factor).

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 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.

7.4.7.5 Triangularization by Givens Rotations This method for triangularization, due to Givens [20], uses a *plane rotation matrix* $T_{ij}(\theta)$ of the following form:

$$T_{ij}(\theta) = \begin{array}{c} \\ \\ j \\ \\ \\ \\ i \\ \\ \\ \end{array} \begin{array}{c} \begin{array}{ccccccccccc} & & & j & & & & i & & & \end{array} \\ \begin{pmatrix} 1 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ 0 & \cdots & 1 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & \cos(\theta) & 0 & \cdots & 0 & \sin(\theta) & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 0 & 1 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ \vdots & & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & & \vdots \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 1 & 0 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & -\sin(\theta) & 0 & \cdots & 0 & \cos(\theta) & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 1 & \cdots & 0 \\ \vdots & & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 1 \end{pmatrix} \end{array}. \tag{7.53}$$

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 7.3. 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 7.7 **(2×2 example)** Consider the problem of upper triangularizing the 2×3 symbolic matrix

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} \tag{7.54}$$

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 Á can be multiplied again on the right by the Givens rotation matrix $T_{13}(\acute{\theta})$ to yield yet a second intermediate matrix form

$$
\begin{aligned}
\breve{A}(\grave{\theta}) &= AT_{23}(\theta)T_{13}(\grave{\theta}) \\
&= \begin{bmatrix} a_{11} & \acute{a}_{12} & \acute{a}_{13} \\ a_{21} & 0 & \acute{a}_{23} \end{bmatrix} \begin{bmatrix} \cos(\grave{\theta}) & & \sin(\grave{\theta}) \\ 0 & 1 & 0 \\ -\sin(\grave{\theta}) & 0 & \cos(\grave{\theta}) \end{bmatrix} \\
&= \begin{bmatrix} \grave{a}_{11} & \acute{a}_{12} & \grave{a}_{13} \\ 0 & 0 & \grave{a}_{23} \end{bmatrix}
\end{aligned}
$$

for $\grave{\theta}$ such that

$$
\cos(\grave{\theta}) = \frac{\acute{a}_{23}}{\sqrt{\acute{a}_{21}^2 + \acute{a}_{23}^2}}, \quad \sin(\grave{\theta}) = \frac{\acute{a}_{21}}{\sqrt{\acute{a}_{21}^2 + \acute{a}_{23}^2}}.
$$

A third Givens rotation yields the final matrix form

$$
\begin{aligned}
\breve{A}(\grave{\theta}) &= AT_{23}(\theta)T_{13}(\grave{\theta})T_{12}(\grave{\theta}) \\
&= \begin{bmatrix} \grave{a}_{11} & \acute{a}_{12} & \grave{a}_{13} \\ 0 & 0 & \grave{a}_{23} \end{bmatrix} \begin{bmatrix} \cos(\grave{\theta}) & \sin(\grave{\theta}) & 0 \\ -\sin(\grave{\theta}) & \cos(\grave{\theta}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} 0 & \breve{a}_{12} & \grave{a}_{13} \\ 0 & 0 & \grave{a}_{23} \end{bmatrix}
\end{aligned}
$$

for $\grave{\theta}$ such that

$$
\cos(\grave{\theta}) = \frac{\acute{a}_{12}}{\sqrt{\grave{a}_{11}^2 + \acute{a}_{12}^2}}, \quad \sin(\grave{\theta}) = \frac{\grave{a}_{11}}{\sqrt{\grave{a}_{11}^2 + \acute{a}_{12}^2}}.
$$

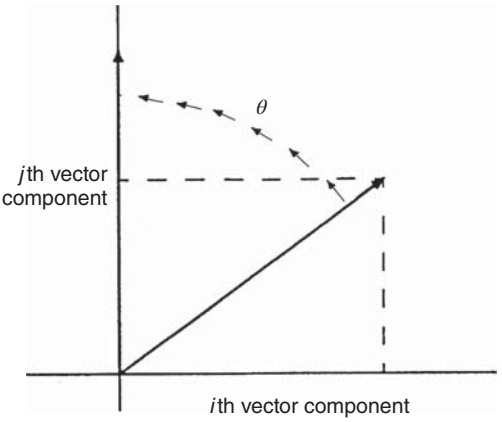


Figure 7.3 Component transformations by plane rotation.

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 7.4 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.

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 listed in the following.

```
Input: A, an - 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;
```

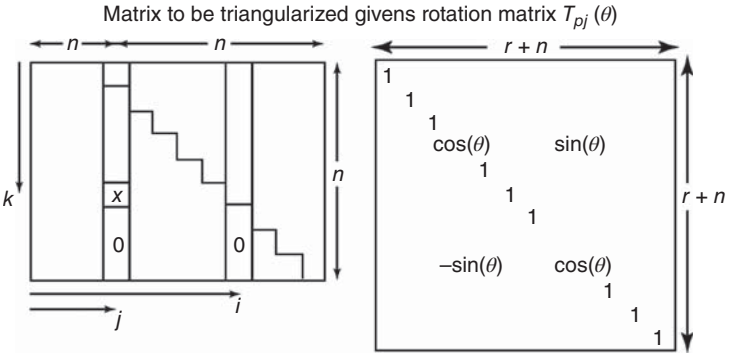


Figure 7.4 Constraints on Givens triangularization order.

```
      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.

7.4.7.6 Triangularization by Householder Reflections This method of triangularization was discovered by Householder [21]. It uses an elementary matrix of the form

$$T(\xi) = I - \frac{2}{\xi^{\mathrm{T}}\xi}\xi\xi^{\mathrm{T}}, \tag{7.55}$$

where ξ 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

$$T(\xi)T^{\mathrm{T}}(\xi) = \left(I - \frac{2}{\xi^{\mathrm{T}}\xi}\xi\xi^{\mathrm{T}}\right)\left(I - \frac{2}{\xi^{\mathrm{T}}\xi}\xi\xi^{\mathrm{T}}\right) \tag{7.56}$$

$$= I - \frac{4}{\xi^{\mathrm{T}}\xi}\xi\xi^{\mathrm{T}} + \frac{4}{(\xi^{\mathrm{T}}\xi)^2}\xi(\xi^{\mathrm{T}}\xi)\xi^{\mathrm{T}} \tag{7.57}$$

$$= I. \tag{7.58}$$

They are called *reflections* because they transform any matrix x into its "mirror reflection" in the plane (or hyperplane[5]) normal to the vector ξ, as illustrated in Figure 7.5

[5]The dimension of the hyperplane normal to the vector ξ will be one less than that of the space containing ξ. When, as in the illustration, ξ is a three-dimensional vector (i.e., the space containing ξ is three-dimensional), the hyperplane normal to ξ is a two-dimensional plane.

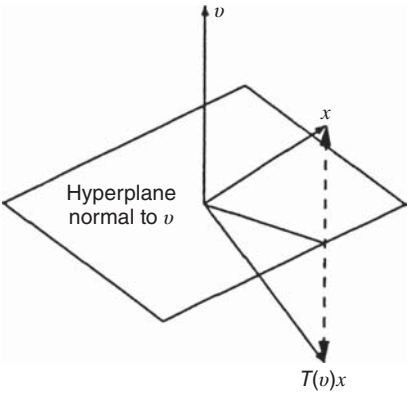


Figure 7.5 Householder reflection of a vector x.

(for three-dimensional ξ and x). By choosing the proper mirror plane, one can place the reflected vector $T(\xi)x$ along any direction whatsoever, including parallel to any coordinate axis.

Example 7.8 (Householder Reflection Along One Coordinate Axis) Let x be any n-dimensional row vector, and let

$$e_k \overset{\text{def}}{=} [0 \quad 0 \quad 0 \quad \ldots \quad 0 \quad \overset{k}{1} \quad 0 \quad \ldots \quad 0]$$

be the kth row of the $n \times n$ identity matrix. If the vector ξ of the Householder reflection $T(\xi)$ is defined as

$$\xi = x^{\mathrm{T}} + \alpha e_k^{\mathrm{T}},$$

where α is a scalar, then the inner products

$$\xi^{\mathrm{T}}\xi = |x|^2 + 2\alpha x_k + \alpha^2,$$
$$x\xi = |x|^2 + \alpha x_k,$$

where x_k is the kth component of x. The Householder reflection $xT(\xi)$ of x will be

$$\begin{aligned} xT(\xi) &= x\left(I - \frac{2}{\xi^{\mathrm{T}}\xi}\xi\xi^{\mathrm{T}}\right) \\ &= x\left(I - \frac{2}{x^{\mathrm{T}}x + 2\alpha x_k + \alpha^2}\xi\xi^{\mathrm{T}}\right) \\ &= x - \frac{2x\xi}{|x|^2 + 2\alpha x_k + \alpha^2}\xi^{\mathrm{T}} \end{aligned}$$

$$
\begin{aligned}
&= x - \frac{2(|x|^2 + \alpha x_k)}{|x|^2 + 2\alpha x_k + \alpha^2}(x + \alpha e_k) \\
&= \left[1 - \frac{2(|x|^2 + \alpha x_k)}{|x|^2 + 2\alpha x_k + \alpha^2}\right] x - \left[\frac{2\alpha(|x|^2 + \alpha x_k)}{|x|^2 + 2\alpha x_k + \alpha^2}\right] e_k \\
&= \left[\frac{\alpha^2 - |x|^2}{|x|^2 + 2\alpha x_k + \alpha^2}\right] x - \left[\frac{2\alpha(|x|^2 + \alpha x_k)}{|x|^2 + 2\alpha x_k + \alpha^2}\right] e_k.
\end{aligned}
$$

Consequently, if one lets

$$\alpha = \mp|x|, \tag{7.59}$$

then

$$xT(\xi) = \pm|x|e_k. \tag{7.60}$$

That is, $xT(\xi)$ 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

$$MT(\xi) = \begin{bmatrix} ZT(\xi) \\ xT(\xi) \end{bmatrix} \tag{7.61}$$

$$= \begin{bmatrix} & & & ZT(\xi) & & \\ 0 & 0 & 0 & \cdots & 0 & |x| \end{bmatrix}, \tag{7.62}$$

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 7.6. 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 reflection matrix $T(\xi)$ 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 $T(\xi)$ 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

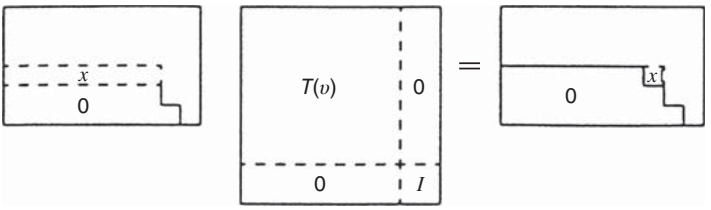


Figure 7.6 Zeroing one subrow with a Householder reflection.

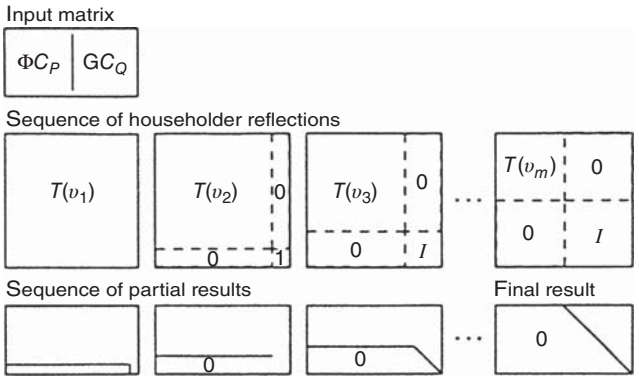


Figure 7.7 Upper triangularization by Householder reflections.

not "unzero" previously zeroed rows below. Together, these features allow the matrix to be triangularized by the sequence of transformations shown in Figure 7.7. Note that the *size* of the Householder transformation shrinks with each step.

Householder Triangularization Algorithm The algorithm listed in the following performs upper triangularization of a rectangular $n \times (n + r)A$ matrix in place using a scratchpad $(n + r)$-vector ξ. 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 Golub and Van Loan [17] 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 7.14.

```
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;
  end;
```

7.5 "SQUARE-ROOT" AND *UD* FILTERS

The so-called "square-root" filtering uses a reformulation of the Riccati equations such that the dependent variable is a generalized 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 7.7.2), for example, performs as well or better, but we are currently not aware of any comparable experience using that method.

7.5.1 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.

7.5.1.1 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 [22]. 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 [23] 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 7.14.

In the case that $m = j$, the summation on the left-hand side of Equation 7.266 has but one term:

$$W_{ij}W_{ij} = \Delta_{ij} - \frac{\mathbf{v}_i\mathbf{v}_j}{R + \sum_{k=1}^{j}\mathbf{v}_k^2}, \tag{7.63}$$

which can be solved in terms of the elements of the upper triangular Cholesky factor W:

$$W_{ij} = \begin{cases} 0, & i > j, \\ \sqrt{\dfrac{R+\sum_{k=1}^{j-1}\mathbf{v}_k^2}{R+\sum_{k=1}^{j}\mathbf{v}_k^2}}, & i = j, \\ \dfrac{-\mathbf{v}_i\mathbf{v}_j}{\left(R+\sum_{k=1}^{j-1}\mathbf{v}_k^2\right)\left(R+\sum_{k=1}^{j}\mathbf{v}_k^2\right)}, & i < j. \end{cases} \tag{7.64}$$

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

$$C_{ij(+)} = \sum_{k=i}^{j} C_{ik(-)}W_{kj} + \text{terms with zero factors} \tag{7.65}$$

$$= C_{ij(-)}W_{jj} + \sum_{k=i}^{j-1} C_{ik(-)}W_{kj} \tag{7.66}$$

$$= C_{ij(-)}\sqrt{\frac{R + \sum_{k=1}^{j-1}\mathbf{v}_k^2}{R + \sum_{k=1}^{j}\mathbf{v}_k^2}} - \sum_{k=i}^{j-1}\frac{C_{ik(-)}\mathbf{v}_k\mathbf{v}_j}{\left(R + \sum_{k=1}^{j-1}\mathbf{v}_k^2\right)\left(R + \sum_{k=1}^{j}\mathbf{v}_k^2\right)} \tag{7.67}$$

$$= \left(R + \sum_{k=1}^{j} \mathbf{v}_k^2 \right)^{-1/2}$$
$$\times \left[C_{ij(-)} \sqrt{R + \sum_{k=1}^{j-1} \mathbf{v}_k^2} - \frac{\mathbf{v}_j \sum_{k=i}^{j-1} C_{ik(-)} \mathbf{v}_k}{\left(R + \sum_{k=1}^{j-1} \mathbf{v}_k^2 \right)^{1/2}} \right]. \tag{7.68}$$

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 7.11. 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 [22].

7.5.1.2 Schmidt Temporal Update

Nonsquare, Nontriangular Cholesky Factor of $P_{k(-)}$ If C_P is a generalized Cholesky factor of $P_{k-1(+)}$ and C_Q is a generalized Cholesky factor of Q_k, then the partitioned $n \times (n + q)$ matrix

$$A = [G_k C_Q \quad | \quad \Phi_k C_P] \tag{7.69}$$

has the $n \times n$ symmetric matrix product value

$$AA^{\mathrm{T}} = [G_{k-1} C_Q | \Phi_{k-1} C_P][G_{k-1} C_Q | \Phi_{k-1} C_P]^{\mathrm{T}} \tag{7.70}$$
$$= \Phi_{k-1} C_P C_P^{\mathrm{T}} \Phi_{k-1}^{\mathrm{T}} + G_{k-1} C_Q C_Q^{\mathrm{T}} G_{k-1}^{\mathrm{T}} \tag{7.71}$$
$$= \Phi_{k-1} P_{k-1(+)} \Phi_{k-1}^{\mathrm{T}} + G_{k-1} Q_{k-1} G_{k-1}^{\mathrm{T}} \tag{7.72}$$
$$= P_{k(-)}. \tag{7.73}$$

That is, A is a nonsquare, nontriangular generalized 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

TABLE 7.11 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 + $n\sqrt{\ }$.

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 7.12. 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 7.4.7. 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 7.13.

TABLE 7.12 Algorithms Performing Matrix Products 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	
$n^2(n+1)/2$	$nr(r+1)/2$

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 it is being triangularized.

Schmidt Temporal Updates Using Householder Reflections The basic Householder triangularization algorithm (see Section 7.4.7.2) operates on a single $n \times (n + r)$ matrix. For the method of Dyer and McReynolds, this matrix is composed of two blocks containing the matrices GC_Q $(n \times r)$ and ΦP $(n \times n)$. 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 7.14. Algorithms for computing GC_Q and ΦP in place were given in the previous subsection.

7.5.2 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

TABLE 7.13 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 + $\frac{1}{2}n(n + 2r + 1)\sqrt{\ }$.

$P = UDU^{\mathrm{T}}$, and the corresponding Thornton algorithm for the temporal update of U and D.

7.5.2.1 Bierman UD Observational Update Bierman's algorithm is one of the more stable implementations of the Kalman filter observational update. It is similar 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.

TABLE 7.14 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 $C_{P_{(-)}} C_{P(-)}^{\mathrm{T}} = \Phi C_{P_{(+)}} C_{P_{(+)}}^{\mathrm{T}} \Phi^{\mathrm{T}} + GC_Q C_Q^{\mathrm{T}} G^{\mathrm{T}}$.
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,
            sigma = sigma + B (i,j)*w (j);
        end;
        for j = 1 : k,
            sigma = sigma + A (i,j)*v (j);
        end;
```

TABLE 7.14 (*Continued*)

```
        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.

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:

$$P_{(-)} \stackrel{\text{def}}{=} U_{(-)}D_{(-)}U_{(-)}^{\mathrm{T}}, \tag{7.74}$$

$$P_{(+)} \stackrel{\text{def}}{=} U_{(+)}D_{(+)}U_{(+)}^{\mathrm{T}}, \tag{7.75}$$

$$U_{(+)}D_{(+)}U_{(+)}^{\mathrm{T}} = U_{(-)}D_{(-)}U_{(-)}^{\mathrm{T}} \tag{7.76}$$

$$- \frac{U_{(-)}D_{(-)}U_{(-)}^{\mathrm{T}}H^{\mathrm{T}}HU_{(-)}D_{(-)}}{R + HU_{(-)}D_{(-)}U_{(-)}^{\mathrm{T}}H^{\mathrm{T}}} U_{(-)}^{\mathrm{T}}$$

$$= U_{(-)}D_{(-)}U_{(-)}^{\mathrm{T}} - \frac{U_{(-)}D_{(-)}\mathbf{v}\mathbf{v}^{\mathrm{T}}D_{(-)}U_{(-)}^{\mathrm{T}}}{R + \mathbf{v}^{\mathrm{T}}D_{(-)}\mathbf{v}} \tag{7.77}$$

$$= U_{(-)}\left[D_{(-)} - \frac{D_{(-)}\mathbf{v}\mathbf{v}^{\mathrm{T}}D_{(-)}}{R + \mathbf{v}^{\mathrm{T}}D_{(-)}\mathbf{v}}\right] U_{(-)}^{\mathrm{T}}, \tag{7.78}$$

where

$$\mathbf{v} = U_{(-)}^{\mathrm{T}}H^{\mathrm{T}} \tag{7.79}$$

is an n-vector and n is the dimension of the state vector.

Equation 7.78 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

$$D_{(-)} - D_{(-)}\mathbf{v}[\mathbf{v}^{\mathrm{T}}D_{(-)}\mathbf{v} + R]^{-1}\mathbf{v}^{\mathrm{T}}D_{(-)} = BD_{(+)}B^{\mathrm{T}}, \tag{7.80}$$

then $D_{(+)}$ would be the a posteriori D factor of P because the resulting equation

$$U_{(+)}D_{(+)}U_{(+)}^{\mathrm{T}} = U_{(-)}\{BD_{(+)}B^{\mathrm{T}}\}U_{(-)}^{\mathrm{T}} \tag{7.81}$$

$$= \{U_{(-)}B\}D_{(+)}\{U_{(-)}B\}^{\mathrm{T}} \tag{7.82}$$

can be solved for the a posteriori U factor as

$$U_{(+)} = U_{(-)}B. \tag{7.83}$$

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 [24] 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 Reference 24. 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 7.15. 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.

7.5.2.2 Thornton UD Temporal Update This UD factorization of the temporal update in the discrete-time Riccati equation is due to Thornton [25]. It is also called *modified weighted Gram–Schmidt* (MWGS) orthogonalization.[6] It uses a factorization algorithm due to Björck [26] that is actually quite different from the conventional Gram–Schmidt orthogonalization algorithm and more robust against roundoff errors. However, the algebraic properties of Gram–Schmidt orthogonalization are useful for deriving the factorization.

[6]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 orthogonalization*. These algorithmic methods were derived by Jorgen Pedersen Gram (1850–1916) and Erhard Schmidt (1876–1959).

TABLE 7.15 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.

Gram–Schmidt orthogonalization is an algorithm for finding a set of n mutually orthogonal m-vectors $b_1, b_2, b_3, \ldots, b_m$ that are linear combinations of a set of n *linearly independent* m-vectors $a_1, a_2, a_3, \ldots, a_m$. That is, the inner products

$$b_i^{\mathrm{T}} b_j = \begin{cases} |b_i|^2 & \text{if } i = j, \\ 0 & \text{if } i \neq j. \end{cases} \tag{7.84}$$

The Gram–Schmidt algorithm defines a unit lower[7] triangular $n \times n$ matrix L such that $A = BL$, where

$$A = \begin{bmatrix} a_1 & a_2 & a_3 & \dots & a_n \end{bmatrix} \tag{7.85}$$

$$= BL \tag{7.86}$$

$$= \begin{bmatrix} b_1 & b_2 & b_3 & \dots & b_n \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ \ell_{21} & 1 & 0 & \dots & 0 \\ \ell_{31} & l_{32} & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \ell_{n1} & \ell_{n2} & \ell_{n3} & \dots & 1 \end{bmatrix}, \tag{7.87}$$

where the vectors b_i are column vectors of B and the matrix product

$$B^{\mathrm{T}}B = \begin{bmatrix} |b_1|^2 & 0 & 0 & \dots & 0 \\ 0 & |b_2|^2 & 0 & \dots & 0 \\ 0 & 0 & |b_3|^2 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & |b_n|^2 \end{bmatrix} \tag{7.88}$$

$$= \mathrm{diag}_{1 \le i \le n}\{|b_i|^2\} \tag{7.89}$$

$$= D_\beta, \tag{7.90}$$

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* $w_1, w_2, w_3, \dots, w_m$ if the *weighted* inner product

$$\sum_{i=1}^{m} x_i w_i y_i = x^{\mathrm{T}} D_w y \tag{7.91}$$

$$= 0, \tag{7.92}$$

where the diagonal *weighting matrix*

$$D_w = \mathrm{diag}_{1 \le i \le n}\{w_i\}. \tag{7.93}$$

[7] In its original form, the algorithm produced a unit upper triangular matrix U by processing the a_i in the order $i = 1, 2, 3, \dots, n$. 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.

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:

$$b_i^{\mathrm{T}} D_w b_j = \begin{cases} \beta_i > 0 & \text{if } i = j, \\ 0 & \text{if } i \neq j. \end{cases} \tag{7.94}$$

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

$$B_w^{\mathrm{T}} D_w B_w = \mathrm{diag}_{1 \leq i \leq n}\{\beta_i\}, \tag{7.95}$$

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 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

$$A = \begin{bmatrix} U_{k-1(+)}^{\mathrm{T}} \Phi_k^{\mathrm{T}} \\ G_k^{\mathrm{T}} \end{bmatrix}, \tag{7.96}$$

$$D_w = \begin{bmatrix} D_{k-1(+)} & 0 \\ 0 & D_{Q_k} \end{bmatrix}, \tag{7.97}$$

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

$$A = BL, \tag{7.98}$$

$$L^{\mathrm{T}} D_\beta L = L^{\mathrm{T}} B^{\mathrm{T}} D_w B L \tag{7.99}$$

$$= (BL)^{\mathrm{T}} D_w B L \tag{7.100}$$

$$= A^{\mathrm{T}} D_w A \tag{7.101}$$

$$= \begin{bmatrix} \Phi_{k-1} U_{k-1(+)} & G_{k-1} \end{bmatrix} \begin{bmatrix} D_{k-1(+)} & 0 \\ 0 & D_{Q_{k-1}} \end{bmatrix} \begin{bmatrix} U_{k-1(+)}^{\mathrm{T}} \Phi_{k-1}^{\mathrm{T}} \\ G_{k-1}^{\mathrm{T}} \end{bmatrix} \tag{7.102}$$

$$= \Phi_{k-1} U_{k-1(+)} D_{k-1(+)} U_{k-1(+)}^{\mathrm{T}} \Phi_{k-1}^{\mathrm{T}} + G_{k-1} D_{Q_{k-1}} G_{k-1}^{\mathrm{T}} \tag{7.103}$$

$$= \Phi_{k-1} P_{k-1(+)} \Phi_{k-1}^{\mathrm{T}} + Q_{k-1} \tag{7.104}$$

$$= P_{k(-)}. \tag{7.105}$$

Consequently, the factors

$$U_{k(-)} = L^{\mathrm{T}}, \tag{7.106}$$

$$D_{k(-)} = D_{\beta} \tag{7.107}$$

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 7.16 includes the computational complexity for the UD decomposition of Q as $U_Q \acute{Q} U_Q^{\mathrm{T}}$ for $\acute{Q}$ diagonal [$\frac{1}{6}p(p-1)(p+4)$ flops] plus the computational complexity for multiplying G by the resulting $p \times p$ unit upper triangular factor U_Q [$\frac{1}{2}np(p-1)$ flops] to obtain $\acute{G}$.

The algorithm listed in Table 7.16 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 ΦU in Place The complexity formulas in Table 7.16 also include the computations required for forming the product ΦU of the $n \times n$ state-transition matrix Φ and the $n \times n$ unit upper triangular matrix U. This matrix product can be performed in place—overwriting Φ with the product ΦU—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.

7.6 *SIGMARHO* FILTERING

Riccati equation implementations using factors of the covariance matrix P not only improve numerical stability of the transformed Riccati equation but also guarantee symmetry and nonnegative definiteness of the resulting covariance matrix.

TABLE 7.16 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	
ΦU 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 $\acute{U}\acute{D}\acute{U}^{\mathrm{T}} = \Phi U D U^{\mathrm{T}} \Phi^{\mathrm{T}} + GQG^{\mathrm{T}}$.

```
for i = n : -1:1,
  sigma = 0;
  for j = 1 : n,
    sigma = sigma + Phi U (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 + Phi U (i, k)*D (k, k)*Phi U (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,
      Phi U (j, k) = Phi U (j, k) - U (j, i)*Phi U (i, k);
    end;
    for k = 1 : p,
      G (j, k) = G (j, k) - U (j, i)*G (i, k);
    end;
  end;
end;
```

(*continued*)

TABLE 7.16 (*Continued*)

Symbol	Description
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.

But these factored approaches have their own drawbacks, the major one being that the factors provide little insight about the nature of the estimation uncertainties. It then becomes necessary to recompute the covariance matrix P from its factors for the purposes of monitoring and assessing estimation uncertainty in familiar terms. Even after P has been computed, it is commonly processed further to calculate the standard deviations σ_i and correlation coefficients ρ_{ij} of state estimation uncertainties to better understand what is going on, and why. These statistical parameters are particularly important during the research, development, testing, and evaluation phases of systems with embedded Kalman filters. They are useful not only for diagnosing filter performance but also for employing corrective adaptation of the filter parameters.

A second problem—from the standpoint of implementation in faster fixed-point arithmetic—is the potentially unconstrained dynamic range of the variables of the factored covariance matrix. Even though the "Bierman rule"[8] suggests a reduction in word-length requirements, the potential dynamic ranges of the variables in the factored Riccati equations are beyond what is needed for straightforward fixed-point implementations.

The "*sigmaRho*" implementation of Grewal and Kain [27] bypasses these issues by propagating the σ_i and ρ_{ij} (hence the name) directly, which facilitates implementation of adaptive measures by directly providing the essential performance diagnostic variables.

Perhaps the most promising attribute of the *sigmaRho* filter is that it is well scaled for implementation in faster fixed-point arithmetic for implementation in specialized high speed digital signal processors. As system-on-chip (SoC) embedded solutions become more mainstream for algorithmic solutions such as the "software radio," high speed Kalman filter implementations are expected to take on greater importance.

The *sigmaRho* implementation equations include three levels of scaling, so they will be derived here in stages:

[8]The claim by the late Gerald J. Bierman (1941–1987) that "square-root" Kalman filtering methods provide "the same precision with half as many bits."

1. Define the basic filter covariance variables, as presented in Chapter 3.
2. Redefine the state variables as the ratios of the standard Kalman filter state variables and their respective standard deviations.
3. Derive the dynamics of these basic state variables, standard deviations, and correlation coefficients in continuous time.
4. Scale the standard deviations for fixed-point representation, and derive the continuous dynamic model for the resulting variables.
5. Derive the equivalent dynamic model in discrete time.
6. Scale the state variables for implementation in fixed-point arithmetic, and derive the appropriate dynamic model for the rescaled variables.
7. Derive the discrete-time measurement update equations for this scaled-state model.

The last two of these stages provide the final implementation equations, using the results of stages 1–5 for their derivations.

7.6.1 Sigma and Rho

Standard deviations (σ_i) and correlation coefficients (ρ_{ij}) are defined in Section 3.3.3.6, where it is shown that a covariance matrix P can be factored as

$$P = D_\sigma C_\rho D_\sigma \tag{7.108}$$

$$= \begin{bmatrix} \sigma_1^2 & \rho_{12}\sigma_1\sigma_2 & \rho_{13}\sigma_1\sigma_3 & \cdots & \rho_{1n}\sigma_1\sigma_n \\ \rho_{21}\sigma_2\sigma_1 & \sigma_2^2 & \rho_{23}\sigma_2\sigma_3 & \cdots & \rho_{2n}\sigma_2\sigma_n \\ \rho_{31}\sigma_3\sigma_1 & \rho_{32}\sigma_3\sigma_2 & \sigma_3^2 & \cdots & \rho_{3n}\sigma_3\sigma_n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho_{n1}\sigma_n\sigma_1 & \rho_{n2}\sigma_n\sigma_2 & \rho_{n3}\sigma_n\sigma_3 & \cdots & \sigma_n^2 \end{bmatrix} \tag{7.109}$$

$$D_\sigma = \text{diag}[\sigma_1,\ \sigma_2,\ \sigma_3,\ \ldots,\ \sigma_n] \tag{7.110}$$

$$= \begin{bmatrix} \sigma_1 & 0 & 0 & \cdots & 0 \\ 0 & \sigma_2 & 0 & \cdots & 0 \\ 0 & 0 & \sigma_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \sigma_n \end{bmatrix} \text{(diagonal matrix)} \tag{7.111}$$

$$C_\rho = \begin{bmatrix} 1 & \rho_{12} & \rho_{13} & \cdots & \rho_{1n} \\ \rho_{21} & 1 & \rho_{23} & \cdots & \rho_{2n} \\ \rho_{31} & \rho_{32} & 1 & \cdots & \rho_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho_{n1} & \rho_{n2} & \rho_{n3} & \cdots & 1 \end{bmatrix} \text{(correlation matrix)} \tag{7.112}$$

$$\rho_{ij} = \rho_{ji}, \tag{7.113}$$

where the σ_i are the *standard deviations* and the ρ_{ij} are the *correlation coefficients*. That is,

$$\sigma_i \stackrel{\text{def}}{=} \sqrt{p_{ii}} \tag{7.114}$$

is the standard deviation of the uncertainty of the ith state variable, ρ_{ij} is the correlation coefficient between the ith and jth state variables, and

$$-1 \leq \rho_{ij} \leq +1. \tag{7.115}$$

The values of the σ_i have the same units as the corresponding state vector components $\hat{x}_i$, and they provide an indication of how well each of the state variables is being estimated. The value of a correlation coefficient ρ_{ij}, if it is near $+1$ or -1, indicates that the uncertainties in the ith and jth state variables are closely linked statistically, a condition which might be improved by changing the measurements that are being used. For example, a sensor measuring only $x_i + x_j$ would tend to make $\rho_{ij} \to -1$, which can be countered by adding a sensor measuring $x_i - x_j$.

In all implementations except *sigmaRho,* computation of the σ_i and the ρ_{ij} requires calculations beyond those necessary for computing P or its factors.

7.6.2 Basic Dynamic Model in Continuous Time

7.6.2.1 Basic State Variables Basic *sigmaRho* state variables x_i' are normalized by dividing the standard state variables x_i of the Kalman filter model by their respective standard deviations of estimation uncertainty:

$$x'(t) \stackrel{\text{def}}{=} D_{\sigma(t)}^{-1} x(t), \tag{7.116}$$

$$x_i'(t) = \frac{x_i(t)}{\sigma_i(t)}, 1 \leq i \leq n, \tag{7.117}$$

where both the x_i and their respective standard deviations σ_i will be functions of time.

7.6.2.2 Basic State Dynamics The dynamics of the standard Kalman filter model state variable x are assumed to be defined by a model of the sort

$$\dot{x}(t) = f(x) + w(t), \tag{7.118}$$

where $f(x)$ is continuously differentiable and time invariant and $\{w(t)\}$ is a zero-mean white noise process with known constant covariance Q.

From Equation 7.116, the time derivative of the state variable x' will now involve time derivatives of x and σ. Using Equation 7.118, the time derivatives of the components x'_i can then be derived as

$$\frac{d}{dt}x'_i = \frac{d}{dt}\frac{x_i}{\sigma_i} \tag{7.119}$$

$$= \frac{\dot{x}_i}{\sigma_i} - x_i \frac{\dot{\sigma}_i}{\sigma_i^2} \tag{7.120}$$

$$= \frac{f_i(x)}{\sigma_i} - x'_i \frac{\dot{\sigma}_i}{\sigma_i}, \tag{7.121}$$

$$\frac{d}{dt}x' = D_\sigma^{-1}[f(D_\sigma x') - D_{x'}\dot{\sigma}], \tag{7.122}$$

where f_i is the ith component of the vector-valued function f and $\dot{\sigma}_i$ will be derived in the next subsubsection.

7.6.2.3 Basic Covariance Dynamics The dynamics of P will be replaced by the dynamics of the σ_i and ρ_{ij}, assuming the dynamics of P to be adequately modeled by the linearized Riccati equation:

$$\dot{P} = FP + PF^{\mathrm{T}} + Q \tag{7.123}$$

$$F \stackrel{\text{def}}{=} \frac{\partial}{\partial x} f(x) \tag{7.124}$$

$$= \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial x_3} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \frac{\partial f_3}{\partial x_1} & \frac{\partial f_3}{\partial x_2} & \frac{\partial f_3}{\partial x_3} & \cdots & \frac{\partial f_3}{\partial x_n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \frac{\partial f_n}{\partial x_3} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}. \tag{7.125}$$

The dynamics of the σ_i and ρ_{ij} can then be derived from Equation 7.123, using the factorization of Equation 7.108. The variable p_{ij} is defined as the element in the ith row and jth column of P. From Equation 7.108, its time derivative

$$\dot{P}_{ij} = \frac{d}{dt}[\rho_{ij}\sigma_i\sigma_j] \tag{7.126}$$

$$= \dot{\rho}_{ij}\sigma_i\sigma_j + \rho_{ij}\dot{\sigma}_i\sigma_j + \rho_{ij}\sigma_i\dot{\sigma}_j. \tag{7.127}$$

In the case that $i = j$, however,

$$\rho_{ii} = 1 \tag{7.128}$$

$$\dot{\rho}_{ii} = 0 \tag{7.129}$$

$$\dot{P}_{ii} = 2\ \dot{\sigma}_i \sigma_i, \tag{7.130}$$

and, from Equation 7.123,

$$\dot{P}_{ii} = \sum_{j=1}^{n} [f_{ij}\rho_{ji}\sigma_j\sigma_i + f_{ij}\rho_{ij}\sigma_i\sigma_j] + q_{ii} \tag{7.131}$$

$$= 2\ \sum_{j=1}^{n} f_{ij}\rho_{ji}\sigma_j\sigma_i + q_{ii}, \tag{7.132}$$

where f_{ij} is the element in the ith row and jth column of F and q_{ij} is the element in the ith row and jth column of Q.

Next, from Equation 7.130,

$$\dot{\sigma}_i = \sum_{j=1}^{n} f_{ij}\rho_{ji}\sigma_j + \frac{q_{ii}}{2\sigma_i} \tag{7.133}$$

$$= \left\{ FD_\sigma C_\rho + \frac{1}{2} D_\sigma Q \right\}_{ii}, \tag{7.134}$$

which provides formulas for the dynamics of σ_i in terms of F, Q, and the σ_j and ρ_{ij} for $1 \leq j \leq n$.

7.6.2.4 Auxilliary Variables Formulas for the dynamics of x' and the ρ_{ij} can be tidied up a bit by using the intermediary $n \times n$ matrix M

$$M \stackrel{\text{def}}{=} D_\sigma^{-1} F D_\sigma C_\rho, \tag{7.135}$$

$$\text{with elements } m_{ij} = \frac{1}{\sigma_i} \sum_{\ell=1}^{n} f_{i\ell}\rho_{\ell j}\sigma_\ell, \tag{7.136}$$

in which case Equation 7.133 becomes

$$\frac{\dot{\sigma}_i}{\sigma_i} = m_{ii} + \frac{q_{ii}}{2\sigma_i^2}, \tag{7.137}$$

$$\text{or } \dot{\sigma}_i = m_{ii}\sigma_i + \frac{q_{ii}}{2\sigma_i}, \tag{7.138}$$

where Equation 7.137 is the form for substitution in Equations 7.121 and 7.138 is the form needed just as a formula for the derivative of σ_i.

TABLE 7.17 Basic *sigmaRho* Dynamics in Continuous Time

State variables	x'	$\stackrel{\text{def}}{=}$	$D_\sigma^{-1}x$
Auxiliary variables	M	$=$	$D_\sigma^{-1}FD_\sigma C_\rho$
σ dynamics	$\dot{\sigma}_i$	$=$	$m_{ii}\sigma_i + \frac{q_{ii}}{2\sigma_i}$
ρ dynamics	$\dot{\rho}_{ij}$	$=$	$-\left[\frac{\dot{\sigma}_i}{\sigma_i} + \frac{\dot{\sigma}_j}{\sigma_j}\right]\ \rho_{ij} + m_{ij} + m_{ji} + \frac{q_{ij}}{\sigma_i\sigma_j}$
State dynamics	$\dot{x}_i'$	$=$	$\frac{1}{\sigma_i}f_i(D_\sigma x') - x_i'\frac{\dot{\sigma}_i}{\sigma_i}$

Notation:
f_{ij} is the ijth element of the matrix $F = \frac{\partial f(x)}{\partial x}$.
f_i is the ith component of the vector $f(\dot{s})$.

When $i \neq j$, Equations 7.123 and 7.127 will yield the identity

$$\begin{aligned}&\dot{\rho}_{ij}\sigma_i\sigma_j + \rho_{ij}\dot{\sigma}_i\sigma_j + \rho_{ij}\sigma_i\dot{\sigma}_j \\ &= \sum_{\ell=1}^{n}[f_{i\ell}\rho_{\ell j}\sigma_\ell\sigma_j + f_{j\ell}\rho_{i\ell}\sigma_i\sigma_\ell] + q_{ij},\end{aligned} \tag{7.139}$$

which can now be solved for

$$\dot{\rho}_{ij} = -\left[\frac{\dot{\sigma}_i}{\sigma_i} + \frac{\dot{\sigma}_j}{\sigma_j}\right]\ \rho_{ij} + m_{ij} + m_{ji} + \frac{q_{ij}}{\sigma_i\sigma_j}, \tag{7.140}$$

where the expressions in square brackets are results of applying Equation 7.137.

The corresponding equation for the derivative of σ_i is Equation 7.138.

This completes the basic *sigmaRho* dynamic model derivations in continuous time, which are summarized in Table 7.17.

7.6.2.5 Integration Issues For some applications, the numerical integration of state and statistical dynamics can be performed by simple trapezoidal integration. This is the case, for example, when sampling frequencies exceed twice the state dynamic bandwidths.

More modern numerical methods are also available to solve nonlinear differential equations in a generalized way regardless of the system dynamics bandwidths. These methods require a computer procedure for computing the derivatives (provided in Tables 7.17 and 7.18), a value of all dynamic terms (state and statistics) at the start of the integration epoch, and a delta-time over which to perform the propagation. The total integration is performed over the input delta-time (between measurements) but this delta-time is broken into subintervals to enforce user-specified criteria for the accuracy of the integration process. An excellent discussion of various integration methods along with software implementing these methods is found in Reference 28. The Richardson extrapolation [29] as implemented by Stoer and Bulirsch (see, e.g., Reference 30 or section 17.3 on p. 921 of Reference 28) is perhaps the most reliable

TABLE 7.18 Normalized *sigmaRho* Dynamics in Continuous Time

State variables	x^*	$\stackrel{\text{def}}{=}$	$D_{\sigma\text{MAX}}^{-1} D_{\sigma}^{-1} x$
Standard deviations	σ'	$=$	$D_{\sigma\text{MAX}}^{-1} \sigma$
Auxiliary variables	F'	$\stackrel{\text{def}}{=}$	$D_{\sigma\text{MAX}}^{-1} F D_{\sigma\text{MAX}}$
	f'_{ij}	$=$	$f_{ij} \sigma_{j\text{MAX}} / \sigma_{i\text{MAX}}$
	Q'	$\stackrel{\text{def}}{=}$	$D_{\sigma\text{MAX}}^{-1} Q D_{\sigma\text{MAX}}^{-1}$
	q'_{ij}	$=$	$q_{ij} / \sigma_{i\text{MAX}} / \sigma_{j\text{MAX}}$
	M'	$\stackrel{\text{def}}{=}$	$D_{\sigma'}^{-1} F D_{\sigma'} C_{\rho}$
	m'_{ij}	$=$	$\frac{1}{\sigma'_i} \sum_{\ell=1}^{n} f'_{i\ell} \rho_{\ell j} \sigma'_{\ell}$
σ dynamics	$\frac{\dot{\sigma}'_i}{\sigma'_i}$	$=$	$m'_{ii} + \frac{q'_{ii}}{2\sigma'^2_i}$
ρ dynamics	$\dot{\rho}_{ij}$	$=$	$-\left[\frac{\dot{\sigma}'_i}{\sigma'_i} + \frac{\dot{\sigma}'_j}{\sigma'_j}\right] \rho_{ij} + m'_{ij} + m'_{ji} + \frac{q'_{ij}}{\sigma'_i \sigma'_j}$
State dynamics	$\dot{x}^*_i$	$=$	$f_i(D_{\sigma} D_{\sigma_{\text{MAX}}} x^*) / \sigma_i / \sigma_{i\text{MAX}} - x^*_i \frac{\dot{\sigma}'_i}{\sigma'_i}$

and efficient method for integrating smooth functions. The fourth-order Runge Kutta method with adaptive step size is considered the most reliable method for dynamic systems that may contain nonsmooth functions [28].

A key detractor for use of such numerical integration methods is that the execution times may not be constant even for constant integration intervals. It is often the case that startup transients result in high rates-of-change early on in the filtering history, so that adaptive step size operations produce more integration subintervals. Such variable and/or unpredictable execution time per measurement is undesirable as the filter designer moves from the conceptual design stage to the operation design stage.

7.6.3 Scaling the σ_i

There are still some numerical scaling issues with the basic *sigmaRho* equations of Table 7.17 for high speed fixed-point implementations.

The numerical values of the variables ρ_{ij} will be between -1 and $+1$, which makes them well conditioned for computation and well suited for implementation in fixed-point arithmetic in low cost high speed signal processors. Computation of the m_{ij} is also well-structured for dot-product implementation in specialized signal processors, although the scaling may need some adjusting for fixed-point implementation.

If the estimation problem is sufficiently observable, then the values of the σ_i will be bounded. It is not uncommon in practice that the maximum expected values are used as the initial standard deviations.

In either case, the numerical values of the dynamic variables (except ρ_{ij}) can be also be rescaled to make them lie between 0 and 1—by dividing by the maximum

expected value of the respective σ_i:

$$\sigma_{i\mathrm{MAX}} \stackrel{\mathrm{def}}{=} \max_t[\sigma_i(t)] \tag{7.141}$$

$$x_i^* \stackrel{\mathrm{def}}{=} x_i'/\sigma_{i\mathrm{MAX}} \tag{7.142}$$

$$\sigma_i' \stackrel{\mathrm{def}}{=} \sigma_i/\sigma_{i\mathrm{MAX}} \tag{7.143}$$

$$\tag{7.144}$$

In practice, the $\sigma_{i\mathrm{MAX}}$ would be determined by simulations—with perhaps a little fudging just to make sure that $\sigma_i \leq \sigma_{i\mathrm{MAX}}$ under all expected operating conditions.

This normalization also rescales the state vector to simplify the dynamic model, but that part of the scaling may not be adequate for fixed-point implementation. The resulting normalized *sigmaRho* dynamic equations in continuous time are summarized in Table 7.18

7.6.4 *SigmaRho* Dynamics in Discrete Time

7.6.4.1 Scaling the State Vector for Fixed-point Implementations This first-level normalization is to keep the computed values of the $\sigma_i' \leq 1$, but there is nothing in the resulting model to prevent components of the normalized state vector x^* from exceeding the range $-1 \leq x_i^* \leq +1$ required for fixed-point implementation. However, it is standard practice in fixed-point implementations to redefine the units of the x_i^* to keep them scaled to the arithmetic binary-point limits of the arithmetic processor. This is accomplished by multiplying by a scaling factor λ_i (commonly a power of 2, so that scaling changes only require bit-shifting, not full multiplication) such that the rescaled variable satisfies the scaling constraint $-1 \leq \lambda_i x_i^* \leq +1$.

7.6.4.2 Scaled Discrete-time Model Dynamic variables of the filter in discrete time include the state vector (x) and its covariance matrix of uncertainty (P), which are normally modeled by equations of the sort

$$x_k = \Phi_{k-1}x_{k-1} + w_{k-1} \tag{7.145}$$

$$\mathop{\mathrm{E}}_{w}\left\langle w_i w_j^{\mathrm{T}}\right\rangle = \begin{cases} 0, & i \neq j \\ Q_i, & i = j \end{cases} \tag{7.146}$$

$$P_k = \Phi_{k-1}P_{k-1}\Phi_{k-1}^{\mathrm{T}} + Q_{k-1} \tag{7.147}$$

$$\Phi_{k-1} = \exp\,[(t_k - t_{k-1})F], \tag{7.148}$$

where F is given by Equation 7.124. However, if f is nonlinear, it is also possible (and perhaps desirable) to propagate the estimate forward in time[9] by integration of $\dot{x} = f(x)$.

[9]This approach, which is common for "extended" Kalman filtering, is also extendable to "Unscented" Kalman filtering (see Chapter 8).

7.6.4.3 Discrete-time Dynamics We are sometimes on safer ground starting with familiar dynamic models in continuous time to develop the Kalman filter in discrete time, and Chapter 2 contains formulas for transitioning from models in continuous time to equivalent models in discrete time. However, some of these formulas are not very efficient for applications with time-varying dynamics, because values of the state-transition matrix Φ may need to be calculated in real time. This may involve taking matrix exponentials, which can be risky [31]. If the time steps Δt are sufficiently small, approximations such as

$$\Phi_{k-1} \approx [I + F(t_{k-1})\ \Delta t] \tag{7.149}$$

$$Q_{k-1} \approx Q(t_{k-1})\ \Delta t \tag{7.150}$$

may suffice, or the formula (due to Van Loan [32]):

$$\exp\ \left(\begin{bmatrix} F\left(t_{k-1}\right) & Q(t_{k-1}) \\ 0 & -F^{\mathrm{T}}(t_{k-1}) \end{bmatrix}\ \Delta t \right) \approx \begin{bmatrix} \Psi & \Phi_{k-1}^{-1} Q_k \\ 0 & \Phi_{k-1}^{\mathrm{T}} \end{bmatrix}. \tag{7.151}$$

In practice, it is always necessary to evaluate the validity of these approximations as part of the development process.

7.6.4.4 SigmaRho State Variables in Discrete Time As in the model for continuous time, the *sigmaRho* state variables are normalized with respect to the standard deviations, component by component:

$$x' \stackrel{\text{def}}{=} D_{\sigma(+)}^{-1} x \tag{7.152}$$

$$\{x_k'\}_i = \frac{\{x_k\}_i}{\{\sigma_{k-1(+)}\}_i}, \tag{7.153}$$

where the notation $\{x_k\}_i$ denotes the ith component of x_k, the value of the state vector at discrete time t_k, and $\{\sigma_{k-1(+)}\}_i$ denotes the ith component of the standard deviations after the last measurement.

7.6.4.5 Rescaled Covariance Variables With this change of state variable, the rescaled estimation error

$$\delta x' = D_{\sigma(+)}^{-1} \delta x \tag{7.154}$$

and its covariance matrix

$$P' \stackrel{\text{def}}{=} \mathop{\mathrm{E}}_{\delta x} \langle \delta x' \delta x'^{\mathrm{T}} \rangle \tag{7.155}$$

$$= D_{\sigma(+)}^{-1} \mathop{\mathrm{E}}_{\delta x} \langle \delta x\ \delta x^{\mathrm{T}} \rangle D_{\sigma(+)}^{-1} \tag{7.156}$$

$$= D_{\sigma(+)}^{-1} P D_{\sigma(+)}^{-1} \tag{7.157}$$

$$= C_{\rho(+)}, \tag{7.158}$$

where P is the covariance matrix for the uncertainty of x and $C_{\rho(+)}$ is the corresponding correlation coefficient matrix after the last measurement update.

That is, the covariance matrix for the rescaled state vector is now the a posteriori correlation coefficient matrix.

If the dynamic model function $f(x)$ is time invariant and $\{\sigma_{k-1(+)}\}_i$ remains constant between measurements, the time derivatives

$$\frac{d}{dt}x' = D^{-1}_{\sigma(+)}\frac{d}{dt}x \tag{7.159}$$

$$= D^{-1}_{\sigma(+)}[Fx(t) + w(t)] \tag{7.160}$$

$$\dot{x}'_i = \frac{1}{\sigma_{i(+)}}\left[\sum_{j=1}^{n} f_{ij}x_j + w_i(t)\right] \tag{7.161}$$

$$= \sum_{j=1}^{n} f_{ij}\frac{\sigma_{j(+)}}{\sigma_{i(+)}}x'_j + \frac{w_i(t)}{\sigma_{i(+)}} \tag{7.162}$$

$$= \sum_{j=1}^{n} f'_{ij}x'_j + \frac{w_i(t)}{\sigma_{i(+)}} \tag{7.163}$$

$$f'_{ij} \stackrel{\text{def}}{=} \frac{\sigma_{j(+)}}{\sigma_{i(+)}}f_{ij} \tag{7.164}$$

$$F' = [f'_{ij}], \text{ an } n \times n \text{ matrix} \tag{7.165}$$

$$= D^{-1}_{\sigma(+)}FD_{\sigma(+)} \tag{7.166}$$

$$F = \frac{\partial f(x)}{\partial x}. \tag{7.167}$$

Now, for this rescaled time-invariant system with dynamic coefficient matrix F', we can define a new state-transition matrix for time step Δt in terms of the state transition for F:

$$\Phi' \stackrel{\text{def}}{=} \exp\ [F'\ \Delta t] \tag{7.168}$$

$$= \sum_{k=0}^{\infty}\frac{1}{k!}F'^k\Delta t^k \tag{7.169}$$

$$= \sum_{k=0}^{\infty}\frac{1}{k!}[D^{-1}_{\sigma(+)}FD_{\sigma(+)}]^k\Delta t^k \tag{7.170}$$

$$= \sum_{k=0}^{\infty}\frac{1}{k!}D^{-1}_{\sigma(+)}[F]^kD_{\sigma(+)}\Delta t^k \tag{7.171}$$

$$= D_{\sigma(+)}^{-1} \left[\sum_{k=0}^{\infty} \frac{1}{k!} F^k \Delta t^k \right] D_{\sigma(+)} \tag{7.172}$$

$$= D_{\sigma(+)}^{-1} \exp\ [F\Delta t] D_{\sigma(+)} \tag{7.173}$$

$$= D_{\sigma(+)}^{-1} \Phi D_{\sigma(+)} \tag{7.174}$$

$$\Phi \stackrel{\text{def}}{=} \exp\ [F\ \Delta t]. \tag{7.175}$$

7.6.4.6 Covariance Time Update The a posteriori covariance $P'_{(+)} = C_{\rho(+)}$, and the next a priori covariance

$$P'_{k(-)} = \Phi' C_{\rho(k-1)(+)} \Phi'^{\mathrm{T}} + Q'_{k-1}, \tag{7.176}$$

from which

$$\{P'_{k(-)}\}_{ii} = \left[\frac{\sigma'_{i\ k(-)}}{\sigma'_{i\ (k-1(+)}} \right]^2 \tag{7.177}$$

$$\frac{\sigma'_{k(-)i}}{\sigma'_{i\ (k-1(+)}} = \sqrt{[\Phi' C_{\rho\ k-1(+)} \Phi'^{\mathrm{T}} + Q'_k]_{ii}} \tag{7.178}$$

$$\{P_{k(-)}\}_{ij} = \rho_{k(-)ij} \sigma'_{k(-)i} \sigma'_{k(-)j} \tag{7.179}$$

$$\rho_{k(-)ij} = \left(\frac{\sigma'_{k-1(+)i}}{\sigma'_{k(+)i}} \right) \left(\frac{\sigma'_{k-1(+)j}}{\sigma'_{k(-)j}} \right) \times [\Phi'_{k-1} C_{\rho\ k-1(+)} \Phi'^{\mathrm{T}}_{k-1} + Q'_{k-1}]_{ij}, \tag{7.180}$$

as summarized in Table 7.19.

TABLE 7.19 Normalized *sigmaRho* Dynamics in Discrete Time

Normalized state	x'	$=$	$\lambda D_{\sigma\mathrm{MAX}}^{-1} D_{\sigma'\ k-1(+)}^{-1} x$
Auxilliary variables	λ	$=$	2^p(state scaling)
	Φ'	$=$	$D_{\sigma\mathrm{MAX}}^{-1} D_{\sigma}^{-1} \Phi D_{\sigma} D_{\sigma\mathrm{MAX}}$
	Q'	$=$	$D_{\sigma\mathrm{MAX}}^{-1} D_{\sigma}^{-1} Q D_{\sigma}^{-1} D_{\sigma\mathrm{MAX}}^{-1}$
State dynamics	$x'_{k+1(-)}$	$=$	$\left(\frac{\sigma_{k(+)i}}{\sigma_{k(-)i}} \right) \{\Phi'_k x'_{k(+)}\}_i$
Standard deviations	$\frac{\sigma'_{i\ k+1(-)}}{\sigma'_{i\ (k-1(+)}}$	$=$	$\sqrt{[\Phi' C_{\rho\ k-1(+)} \Phi'^{\mathrm{T}} + Q'_k]_{ii}}$
Correlation coefficient	$\rho_{k(-)ij}$	$=$	$\left(\frac{\sigma'_{k-1(+)i}}{\sigma'_{k(+)i}} \right) \left(\frac{\sigma'_{k-1(+)j}}{\sigma'_{k(-)j}} \right)$ $\times\ [\Phi'_{k-1} C_{\rho\ k-1(+)} \Phi'^{\mathrm{T}}_{k-1} + Q'_{k-1}]_{ij}$

7.6.5 *SigmaRho* Measurement Updates

7.6.5.1 Standard Equations In conventional Kalman filtering, the measurement update at discrete time t_k uses a measurement $z_k = H_k x_{k(-)} + v_k$ to improve the a priori estimate $\hat{x}_{k(-)}$ and obtain a better a posteriori estimate $\hat{x}_{k(+)}$ and its associated a posteriori covariance of estimation uncertainty $P_{k(+)}$, using the Kalman gain and update formulas

$$\overline{K}_k = P_{k(-)} H_k^{\mathrm{T}} \underbrace{[H_k P_{k(-)} H_k^{\mathrm{T}} + R_k]}_{P_{\nu\nu,k}}{}^{-1} \tag{7.181}$$

$$P_{k(+)} = [I - \overline{K}_k H_k] P_{k(-)}, \tag{7.182}$$

$$x_{k(+)} = x_{k(-)} + \overline{K}_k \underbrace{[z_k - H_k x_{k(-)}]}_{\nu_k}, \tag{7.183}$$

where the term in square brackets in Equation 7.183 is the *measurement innovation*[10] ν_k (also called *measurement residual*), representing the information in the matrix that could not be predicted from the estimated state. The term inside square brackets in Equation 7.181 is its theoretical covariance $P_{\nu\nu,k}$, also called the *covariance of innovations* or *covariance of measurement residuals*.

7.6.5.2 Scalar Measurements As we showed for the "square-root" implementations of the Kalman filter, one can always assume that the measurements are scalars, because—if not–they can be easily transformed into independent scalar measurements.[11]

In that case, the measurement sensitivity matrices H_k will be $1 \times n$ row vectors, the Kalman gain $\overline{K}_k$ will be a $n \times 1$ column vector, and the term in square brackets in Equation 7.181 will be a scalar,

$$\sigma_{\nu k}^2 = P_{\nu\nu,k}, \tag{7.184}$$

which transforms a matrix divide problem into a scalar divide problem and greatly simplifies the derivation. Also, the scalar

$$R_k = \sigma_{v,k}^2, \tag{7.185}$$

the variance of measurement noise v_k.

[10] Innovations are often represented by the Greek letter ν (nu) because they represent "what is new" in the measurement. However, the typeset ν is quite similar to the typeset v (measurement noise).

[11] Kaminski et al. [33] have shown that the same ploy speeds up the conventional Kalman filter implementation, as well.

7.6.5.3 Kalman Gain When z_k is a scalar, the Kalman gain formula of Equation 7.181 can be composed from vector expressions, using the decomposition

$$P_{k(-)} = D_{\sigma\ k(-)} C_{\rho\ k(-)} D_{\sigma\ k(-)}, \tag{7.186}$$

so that the vector–matrix products

$$H_k P_{k(-)} = \underbrace{H_k D_{\sigma\ k(-)} C_{\rho\ k(-)}}_{d_k} D_{\sigma\ k(-)} \tag{7.187}$$

$$= d_k D_{\sigma\ k(-)} \tag{7.188}$$

$$d_k \overset{\text{def}}{=} H_k D_{\sigma\ k(-)} C_{\rho\ k(-)} \tag{7.189}$$

$$H_k P_{k(-)} H_k^{\mathrm{T}} = d_k D_{\sigma\ k(-)} H_k^{\mathrm{T}} \tag{7.190}$$

$$\sigma_{\nu\ k}^2 = H_k P_{k(-)} H_k^{\mathrm{T}} + R_k \tag{7.191}$$

$$= d_k D_{\sigma\ k(-)} H_k^{\mathrm{T}} + R_k \tag{7.192}$$

$$K_k = \frac{1}{\sigma_{\nu\ k}^2} D_{\sigma\ k(-)} d_k^{\mathrm{T}}, \tag{7.193}$$

a column vector.

7.6.5.4 Covariance Update Using the decomposition of Equation 7.186, Equation 7.182 becomes

$$D_{\sigma\ k(+)} C_{\rho\ k(+)} D_{\sigma\ k(+)} = D_{\sigma\ k(-)} C_{\rho\ k(-)} D_{\sigma\ k(-)} - K_k H_k P_{k(-)}, \tag{7.194}$$

which—using Equations 7.188 and 7.193—becomes

$$D_{\sigma\ k(+)} C_{\rho\ k(+)} D_{\sigma\ k(+)} = D_{\sigma\ k(-)} C_{\rho\ k(-)} D_{\sigma\ k(-)} - K_k d_k D_{\sigma\ k(-)} \tag{7.195}$$

$$= D_{\sigma\ k(-)} C_{\rho\ k(-)} D_{\sigma\ k(-)} - \frac{1}{\sigma_{\nu\ k}^2} D_{\sigma\ k(-)} d_k^{\mathrm{T}} d_k D_{\sigma\ k(-)} \tag{7.196}$$

$$= D_{\sigma\ k(-)} \left[C_{\rho\ k(-)} - \frac{1}{\sigma_{\nu\ k}^2} d_k^{\mathrm{T}} d_k \right] D_{\sigma\ k(-)} \tag{7.197}$$

$$C_{\rho\ k(+)} = D_{\sigma\ k(+)}^{-1} D_{\sigma\ k(-)} \left[C_{\rho\ k(-)} - \frac{1}{\sigma_{\nu\ k}^2} d_k^{\mathrm{T}} d_k \right] D_{\sigma\ k(-)} D_{\sigma\ k(+)}^{-1}. \tag{7.198}$$

Because the diagonal elements $\rho_{ii} = 1$, the diagonal elements of Equation 7.197 become

$$\sigma^2_{k(+),i} = \sigma^2_{k(+),i}\left[1 - \frac{d^2_{k,i}}{\sigma^2_{\nu\ k}}\right] \tag{7.199}$$

$$\sigma_{k(+),i} = \sigma_{k(+),i}\sqrt{1 - \frac{d^2_{k,i}}{\sigma^2_{\nu\ k}}}. \tag{7.200}$$

The ratios

$$\frac{\sigma_{k(+),i}}{\sigma_{k(+),i}} = \sqrt{1 - \frac{d^2_{k,i}}{\sigma^2_{\nu\ k}}} \tag{7.201}$$

$$= \frac{1}{\sigma_{\nu\ k}}\sqrt{\sigma^2_{\nu\ k} - d^2_{k,i}}, \tag{7.202}$$

are the *standard deviation improvement ratios* from the measurement update.

7.6.5.5 Estimate Update The standard update formula,

$$\hat{x}_{k(+)} = \hat{x}_{k(+)} + K_k[z_k - H_k\hat{x}_{k(-)}], \tag{7.203}$$

7.6.5.6 Normalized Implementation With the scaling of Section 7.6.4.3, the normalized parameters and variables become

$$D_{\sigma' k} = D^{-1}_{\sigma\mathrm{MAX}} D_{\sigma\ k} \tag{7.204}$$

$$H'_k = D_{\sigma' k(-)} H_k \tag{7.205}$$

$$\hat{x}'_k = \lambda D_{\sigma' k}\hat{x}_k, \tag{7.206}$$

and the normalized measurement update implementation equations become

$$d'_k = H'_k D_{\sigma',k(-)} C_{\rho\ k(-)}. \tag{7.207}$$

7.6.6 Efficacy

The *sigmaRho* Kalman filter implementation is still quite new and would benefit from further development and applications experience. Besides its natural heuristic attributes for safe and reliable development of Kalman filter applications, the implementation lends itself to high speed applications where fixed-point arithmetic is required, and potentially to applications where process and measurement noise covariances might be varied for adaptive filtering.

TABLE 7.20 Normalized *SigmaRho* Measurement Update

State variable	x^*	$=$	$\lambda D_{\sigma\text{MAX}}^{-1} D_\sigma^{-1} x$
Auxiliary variables	λ	$\overset{\text{def}}{=}$	2^p(scaling factor)
	H'_k	$=$	$D_{\sigma' k(-)} H_k$
	d'_k	$=$	$H'_k D_{\sigma' k(-)} C_{\rho\ k(-)}$
Innovations covariance	$\sigma^2_{\nu\ k}$	$=$	$d_k D_{\sigma\ k(-)} H_k^{\mathrm{T}} + R_k$
Standard deviation improvement ratio	$\dfrac{\sigma_{k(+),i}}{\sigma_{k(+),i}}$	$=$	$\sqrt{1 - \dfrac{d^2_{k,i}}{\sigma^2_{\nu\ k}}}$
Correlation coefficients	$\rho_{k(+)ij}$	$=$	$\left[\dfrac{\sigma_i'^-}{\sigma_i'^+}\right]\left[\dfrac{\sigma_j'^-}{\sigma_j'^+}\right]\left[\rho_{k(-)ij} - \dfrac{D_i D_j}{\Omega^2}\right]$
State update	$x'_{i(+)}$	$=$	$\left[\dfrac{\sigma_{k(-)i}}{\sigma_i^+}\right]\left\{x'_i(-) + \dfrac{D_i}{\Omega}\left[\dfrac{\lambda z_k - H'_k \sigma'_k x'_k}{\Omega}\right]\right\}$

The *sigmaRho* filter has been implemented for a sampling rate of 74 MHz for signal tracking of binary phase-shift keying (BPSK) in a software receiver [27]. This particular nonlinear application has closed-form solutions for the state-transition matrix and process noise covariance for a four-state *sigmaRho* implementation. The result was well scaled for fast fixed-point implementation. Performance was very good, which is encouraging for Kalman filter implementations at those speeds. Hopefully, this will spark more interest in further development and evaluation of the *sigmaRho* Kalman filter for embedded Kalman filter applications with speed requirements that might otherwise be considered excessive.

7.7 OTHER IMPLEMENTATION METHODS

The main thrust of this chapter is on 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 other 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.

7.7.1 Earlier Implementation Methods

7.7.1.1 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.

Recursive Least Mean Squares This form of the covariance update for recursive least-mean-squares estimation was published by Swerling [34]. Swerling's estimator

TABLE 7.21 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	$2n^3 + n^2 + n + \frac{1}{2}n^2\ell + n\ell^2 + \frac{1}{2}n\ell + \ell^3 + \frac{1}{2}\ell^2 + \ell$

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[12] For the case that the state dimension $n = 1$ and the measurement dimension $\ell = 1$, it can be done with 4 flops. In the case that $n > 1$, the dependence of the computational complexity on ℓ and n is shown in Table 7.21.[13] 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.

7.7.1.2 Kalman Formulation

Data Flows A data flow diagram of the implementation equations defined by Kalman [36] is shown in Figure 7.8. 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.

[12] See Section 7.4.4.7 for an explanation of how computational complexity is determined.

[13] There is an alternative matrix inversion method (due to Strassen [35]) 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.

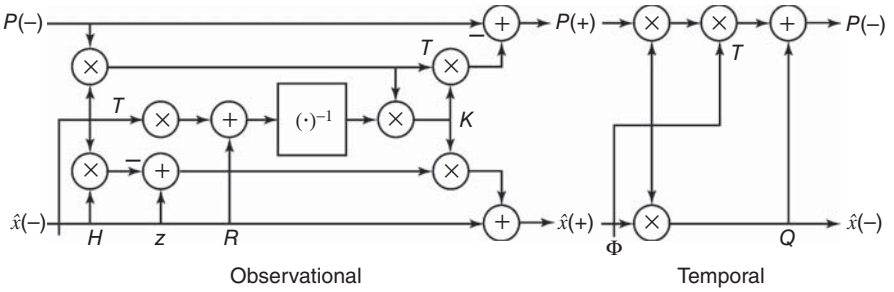


Figure 7.8 Data flows of Kalman update implementations.

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:

$$\overline{K} = P_{(-)}H^{\mathrm{T}}[HP_{(-)}H^{\mathrm{T}} + R]^{-1} \tag{7.208}$$

$$= \{HP_{(-)}\}^{\mathrm{T}}[\{HP_{(-)}\}H^{\mathrm{T}} + R]^{-1}, \tag{7.209}$$

$$P_{(+)} = P_{(-)} - \overline{K}\{HP_{(-)}\}. \tag{7.210}$$

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 number of floating-point arithmetic operations required for the observational update of the Kalman filter, using these methods, is summarized in Table 7.22. Note that the total number of operations required does not increase as n^3, as was the case with the Swerling formulation.

7.7.1.3 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 methods for the observational update (see Battin [37, pp. 338–340] and Potter and Stern [38]).

Potter defined the generalized Cholesky factor of the covariance matrix P as

$$P_{(-)} \stackrel{\text{def}}{=} C_{(-)}C_{(-)}^{\mathrm{T}}, \tag{7.211}$$

$$P_{(+)} \stackrel{\text{def}}{=} C_{(+)}C_{(+)}^{\mathrm{T}}, \tag{7.212}$$

so that the observational update equation

$$P_{(+)} = P_{(-)} - P_{(-)}H^{\mathrm{T}}[HP_{(-)}H^{\mathrm{T}} + R]^{-1}HP_{(-)} \tag{7.213}$$

could be partially factored as

$$C_{(+)}C_{(+)}^{\mathrm{T}} = C_{(-)}C_{(-)}^{\mathrm{T}} - C_{(-)}C_{(-)}^{\mathrm{T}}H^{\mathrm{T}}[HC_{(-)}C_{(-)}^{\mathrm{T}}H^{\mathrm{T}} + R]^{-1}HC_{(-)}C_{(-)}^{\mathrm{T}} \quad (7.214)$$

$$= C_{(-)}C_{(-)}^{\mathrm{T}} - C_{(-)}V[V^{\mathrm{T}}V + R]^{-1}V^{\mathrm{T}}C_{(-)}^{\mathrm{T}} \quad (7.215)$$

$$= C_{(-)}\{I_n - V[V^{\mathrm{T}}V + R]^{-1}V^{\mathrm{T}}\}C_{(-)}^{\mathrm{T}}, \quad (7.216)$$

where

$I_n = \times n$ identity matrix,

$V = C_{(-)}^{\mathrm{T}}H^{\mathrm{T}}$ is an $n \times \ell$ general matrix,

$n =$ dimension of state vector, and

$\ell =$ dimension of measurement vector.

Equation 7.216 contains the unfactored expression $\{I_n - V[V^{\mathrm{T}}V + R]^{-1}V^{\mathrm{T}}\}$. For the case that the measurement is a scalar ($\ell = 1$), Potter was able to factor it in the form

$$I_n - V[V^{\mathrm{T}}V + R]^{-1}V^{\mathrm{T}} = WW^{\mathrm{T}}, \quad (7.217)$$

so that the resulting equation

$$C_{(+)}C_{(+)}^{\mathrm{T}} = C_{(-)}\{WW^{\mathrm{T}}\}C_{(-)}^{\mathrm{T}} \quad (7.218)$$

$$= \{C_{(-)}W\}\{C_{(-)}W\}^{\mathrm{T}} \quad (7.219)$$

could be solved for the a posteriori generalized Cholesky factor of $P_{(+)}$ as

$$C_{(+)} = C_{(-)}W. \quad (7.220)$$

When the measurement is a scalar, the expression to be factored is a symmetric *elementary matrix* of the form[14]

$$I_n - \frac{\mathbf{v}\mathbf{v}^{\mathrm{T}}}{R + |\mathbf{v}|^2}, \quad (7.221)$$

where R is a positive scalar and $\mathbf{v} = C_{(-)}^{\mathrm{T}}H^{\mathrm{T}}$ is a column n-vector.

[14]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 7.5.1.1) finds a triangular factor W, which preserves 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 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*.

TABLE 7.22 Operation Summary for 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	$\frac{3}{2}n^2\ell + \frac{3}{2}n\ell^2 + n\ell + \ell^3 + \frac{1}{2}\ell^2 + \frac{1}{2}\ell$

The formula for the symmetric square root of a symmetric elementary matrix is given in Equation 7.44. For the elementary matrix format in Equation 7.221, the scalar s of Equation 7.44 has the value

$$s = \frac{1}{R + |\mathbf{v}|^2}, \tag{7.222}$$

so that the radicand

$$1 - s|\mathbf{v}|^2 = 1 - \frac{|\mathbf{v}|^2}{R + |\mathbf{v}|^2} \tag{7.223}$$

$$= \frac{R}{R + |\mathbf{v}|^2} \tag{7.224}$$

$$\geq 0 \tag{7.225}$$

because the variance $R \geq 0$. Consequently, the matrix expression 7.221 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 generalized Cholesky factors. That is,

$$(I - s\mathbf{v}\mathbf{v}^{\mathrm{T}}) = (I - \sigma\mathbf{v}\mathbf{v}^{\mathrm{T}})(I - \sigma\mathbf{v}\mathbf{v}^{\mathrm{T}}) \tag{7.226}$$

$$= (I - \sigma\mathbf{v}\mathbf{v}^{\mathrm{T}})(I - \sigma\mathbf{v}\mathbf{v}^{\mathrm{T}})^{\mathrm{T}}. \tag{7.227}$$

Following the approach leading to Equation 7.220, the solution for the a posteriori generalized Cholesky factor $C_{(+)}$ of the covariance matrix P can be expressed as the product

$$C_{(+)}C_{(+)}^{\mathrm{T}} = P_{(+)} \tag{7.228}$$

$$= C_{(-)}(I - s\mathbf{v}\mathbf{v}^{\mathrm{T}})C_{(-)}^{\mathrm{T}} \tag{7.229}$$

$$= C_{(-)}(I - \sigma\mathbf{v}\mathbf{v}^{\mathrm{T}})(I - \sigma\mathbf{v}\mathbf{v}^{\mathrm{T}})^{\mathrm{T}}C_{(-)}^{\mathrm{T}}, \tag{7.230}$$

which can be factored as[15]

$$C_{(+)} = C_{(-)}(I - \sigma \mathbf{v}\mathbf{v}^{\mathrm{T}}) \tag{7.231}$$

with

$$\sigma = \frac{1 + \sqrt{1 - s|\mathbf{v}|^2}}{|\mathbf{v}|^2} \tag{7.232}$$

$$= \frac{1 + \sqrt{R/(R + |\mathbf{v}|^2)}}{|\mathbf{v}|^2}. \tag{7.233}$$

Equations 7.231 and 7.233 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 generalized Cholesky factor C of P in place. This generalized 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.

7.7.1.4 Joseph-Stabilized Implementation This variant of the Kalman filter is due to Joseph [39], who demonstrated improved numerical stability by rearranging the standard formulas for the observational update (given here for scalar measurements) into the formats

$$\acute{z} = R^{-1/2} z, \tag{7.234}$$

$$\acute{H} = \acute{z} H, \tag{7.235}$$

$$\overline{K} = (\acute{H} P_{(-)} \acute{H}^{\mathrm{T}} + 1)^{-1} P_{(-)} \acute{H}^{\mathrm{T}}, \tag{7.236}$$

$$P_{(+)} = (I - \overline{K}\acute{H}) P_{(-)} (I - \overline{K}\acute{H})^{\mathrm{T}} + \overline{K}\,\overline{K}^{\mathrm{T}}, \tag{7.237}$$

taking advantage of partial results and the redundancy due to symmetry. The mathematical equivalence of Equation 7.237 to the conventional update formula for the covariance matrix was shown as Equation 5.18. This formula, by itself, does not 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 [24] that reduces computational complexity by measurement decorrelation (if necessary) and the parsimonious use of partial results. The data flow diagram shown in Figure 7.9 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

[15]Note that as $R \to \infty$ (no measurement), $\sigma \to 2/|v|^2$ and $I - \sigma v v^{\mathrm{T}}$ becomes a Householder matrix.

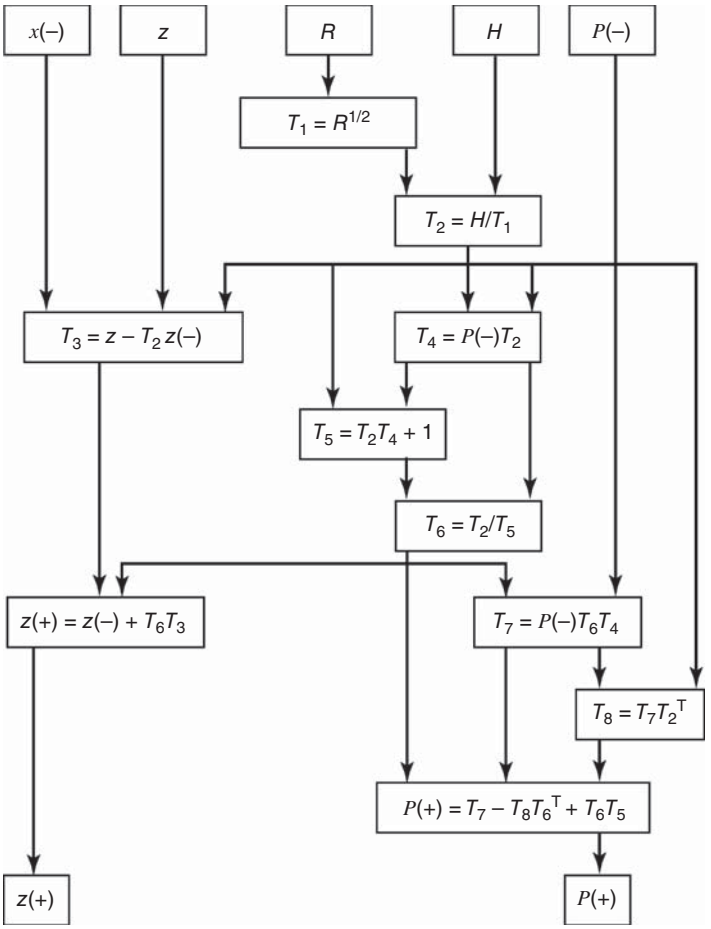


Figure 7.9 Data flow of Bierman–Joseph implementation.

may be implemented in parallel. Intermediate (temporary) results are labeled as $\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_8$, where $\mathcal{F}_6 = \overline{K}$, 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 [24] 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 of components in the measurement vector [24]. However, this formulation does require that R be a diagonal matrix. Otherwise, an 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.

TABLE 7.23 De Vries–Joseph Implementation of Covariance Update

Operation	Complexity
	Without Using Decorrelation
$\mathcal{F}_1 = P_{(-)}H^{\mathrm{T}}$	ℓn^2
$\mathcal{F}_2 = H\mathcal{F}_1 + R$	$n\ell(\ell+1)/2$
$\mathcal{U}\mathcal{D}\mathcal{U}^{\mathrm{T}} = \mathcal{F}_2$	$\frac{1}{6}\ell(\ell+1)(\ell+2)$ (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:	$\frac{2}{3}\ell^3 + \ell^2 - \frac{5}{3}\ell - \frac{1}{2}\ell n + \frac{1}{2}\ell^2 n$
	$\ell \times \{$
$\mathcal{F}_1 = P_{(-)}H^{\mathrm{T}}$	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	$\frac{2}{3}\ell^3 + \ell^2 - \frac{2}{3}\ell + \frac{5}{3}\ell n + \frac{1}{2}\ell^2 n + 2\ell n^2$

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 7.23.

Negative Evaluations of Joseph-Stabilized Implementation In comparative evaluations of several Kalman filter implementations on orbit estimation problems, Thornton and Bierman [14] found that the Joseph-stabilized implementation failed on some ill-conditioned problems for which "square-root" methods performed well.

7.7.2 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 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 Martin Morf and Thomas Kailath [40].

7.7.2.1 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

$$A_k = \begin{bmatrix} GC_{Q(k)} & \Phi_k C_{P(k)} & 0 \\ 0 & H_k C_{P(k)} & C_{R(k)} \end{bmatrix}, \tag{7.238}$$

where $C_{R(k)}$ is a generalized Cholesky factor of R_k and $C_{Q(k)}$ is a generalized Cholesky factor of Q_k. Note that the $(n+m)\times(n+m)$ symmetric product

$$A_k A_k^{\mathrm{T}} = \begin{bmatrix} \Phi_k P_{k(-)} \Phi_k^{\mathrm{T}} + G_k Q_k G_k^{\mathrm{T}} & \Phi_k P_{k(-)} H_k^{\mathrm{T}} \\ H_k P_{k(-)} \Phi_k^{\mathrm{T}} & H_k P_{k(-)} H_k^{\mathrm{T}} \end{bmatrix}. \tag{7.239}$$

Consequently, if A_k is upper triangularized in the form

$$A_k T = C_k \tag{7.240}$$

$$= \begin{bmatrix} 0 & C_{P(k+1)} & \Psi_k \\ 0 & 0 & C_{E(k)} \end{bmatrix} \tag{7.241}$$

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 $C_{E(k)}, \Psi_k$, and $C_{P(k+1)}$ satisfy the equations

$$C_{E(k)} C_{E(k)}^{\mathrm{T}} = H_k P_{k(-)} H_k^{\mathrm{T}} + R_k \tag{7.242}$$

$$= E_k, \tag{7.243}$$

$$\Psi_k \Psi_k^{\mathrm{T}} = \Phi_k P_{k(-)} H_k^{\mathrm{T}} E_k^{-1} H_k P_{k(-)} \Phi_k, \tag{7.244}$$

$$\Psi_k = \Phi_k P_{k(-)} H_k^{\mathrm{T}} C_{E(k)}^{-1}, \tag{7.245}$$

$$C_{P(k+1)} C_{P(k+1)}^{\mathrm{T}} = \Phi_k P_{k(-)} \Phi_k^{\mathrm{T}} + G_k Q_k G_k^{\mathrm{T}} - \Psi_k \Psi_k^{\mathrm{T}} \tag{7.246}$$

$$= P_{k+1(-)}, \tag{7.247}$$

and the Kalman gain can be computed as

$$\overline{K}_k = \Psi_k C_{E(k)}^{-1}. \tag{7.248}$$

The computation of C_k from A_k can be done by Householder or Givens triangularization.

7.7.2.2 Combined Updates of UD Factors This implementation uses the *UD* factors of the covariance matrices *P, R*, and *Q*,

$$P_k = U_{P(k)} D_{P(k)} U_{P(k)}^{\mathrm{T}}, \tag{7.249}$$

$$R_k = U_{R(k)} D_{R(k)} U_{R(k)}^{\mathrm{T}}, \tag{7.250}$$

$$Q_k = U_{Q(k)} D_{Q(k)} U_{Q(k)}^{\mathrm{T}}, \tag{7.251}$$

in the partitioned matrices

$$B_k = \begin{bmatrix} G U_{Q(k)} & \Phi_k U_{P(k)} & 0 \\ 0 & H_k U_{P(k)} & U_{R(k)} \end{bmatrix}, \tag{7.252}$$

$$D_k = \begin{bmatrix} D_{Q(k)} & 0 & 0 \\ 0 & D_{P(k)} & 0 \\ 0 & 0 & D_{R(k)} \end{bmatrix}, \tag{7.253}$$

which satisfy the equation

$$B_k D_k B_k^{\mathrm{T}} = \begin{bmatrix} \Phi_k P_{k(-)} \Phi_k^{\mathrm{T}} + G_k Q_k G_k^{\mathrm{T}} & \Phi_k P_{k(-)} H_k^{\mathrm{T}} \\ H_k P_{k(-)} \Phi_k^{\mathrm{T}} & H_k P_{k(-)} H_k^{\mathrm{T}} \end{bmatrix}. \tag{7.254}$$

The MWGS orthogonalization of the rows of B_k with respect to the weighting matrix D_k will yield the matrices

$$\acute{B}_k = \begin{bmatrix} U_{P(k+1)} & U_{\Psi(k)} \\ 0 & U_{E(k)} \end{bmatrix}, \tag{7.255}$$

$$\acute{D}_k = \begin{bmatrix} D_{P(k+1)} & 0 \\ 0 & D_{E(k)} \end{bmatrix}, \tag{7.256}$$

where $U_{P(k+1)}$ and $D_{P(k+1)}$ are the *UD* factors of $P_{k+1(-)}$, and

$$U_{\Psi(k)} = \Phi_k P_{k(-)} H_k^{\mathrm{T}} U_{E(k)}^{-\mathrm{T}} D_{E(k)}^{-1} \tag{7.257}$$

$$= \overline{K}_k U_{E(k)}. \tag{7.258}$$

Consequently, the Kalman gain

$$\overline{K}_k = U_{\Psi(k)} U_{E(k)}^{-1} \tag{7.259}$$

can be computed as a by-product of the MWGS procedure, as well.

7.7.3 Information Filtering

7.7.3.1 Information Matrix of an Estimate The inverse of the covariance matrix of estimation uncertainty is called the *information matrix*:[16]

$$Y \stackrel{\text{def}}{=} P^{-1}. \tag{7.260}$$

Implementations using Y (or its generalized Cholesky factors) rather than P (or its generalized Cholesky factors) are called *information filters*. (Implementations using P are also called *covariance filters*.

7.7.3.2 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

$$Y_0 = 0, \tag{7.261}$$

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

$$Y_0 = \sum_i \lambda_i e_i e_i^{\mathrm{T}}, \tag{7.262}$$

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.

[16]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.

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.

7.7.3.3 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*

$$d \overset{\text{def}}{=} Yx, \tag{7.263}$$

and those that use its generalized Cholesky factors C_Y such that

$$C_Y C_Y^{\mathrm{T}} = Y \tag{7.264}$$

use the "square-root" information state

$$s \overset{\text{def}}{=} C_Y x. \tag{7.265}$$

TABLE 7.24 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

$$A_k \stackrel{\mathrm{def}}{=} \Phi_k^{-\mathrm{T}} Y_{k(+)} \Phi_k^{-1}$$
$$Y_{k+1(-)} = \{I - A_k G_k [G_k^{\mathrm{T}} A_k G_k + Q_k^{-1}]^{-1} G_k^{\mathrm{T}}\} A_k$$
$$\hat{d}_{k+1(-)} = \{I - A_k G_k [G_k^{\mathrm{T}} A_k G_k + Q_k^{-1}]^{-1} G_k^{\mathrm{T}}\} \Phi_k^{-\mathrm{T}} \hat{d}_{k(+)}$$

7.7.3.4 Information Filter Implementation The implementation equations for the "straight" information filter (i.e., using Y, rather than its generalized Cholesky factors) are shown in Table 7.24. These can be derived from the Kalman filter equations and the definitions of the information matrix and information state. Note the similarities in form between these equations and the Kalman filter equations, with respective observational and temporal equations switched.

7.7.3.5 "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 [11], using the "square-root" least-squares methods (triangularization) developed by Golub [41] and applied to sequential least-squares estimation by Lawson and Hanson [42]. The form developed by Dyer and McReynolds is shown in Table 7.25.

7.8 SUMMARY

Although Kalman filtering has been called "ideally suited to digital computer implementation" [43], 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.

TABLE 7.25 Square-Root Information Filter Using Triangularization

Observational update

$$\begin{bmatrix} C_{Y_{k(-)}} & H_k^{\mathrm{T}} C_{R_k^{-1}} \\ \hat{s}_{k(-)}^{\mathrm{T}} & z_k^{\mathrm{T}} C_{R_k^{-1}} \end{bmatrix} \mathrm{T}_{\mathrm{obs}} = \begin{bmatrix} C_{Y_{k(+)}} & 0 \\ \hat{s}_{k(+)}^{\mathrm{T}} & \epsilon \end{bmatrix}$$

Temporal update

$$\begin{bmatrix} C_{Q_k^{-1}} & -G_k \Phi_k^{-\mathrm{T}} C_{Y_{k(+)}} \\ 0 & \Phi_k^{-\mathrm{T}} C_{Y_{k(+)}} \\ 0 & \hat{s}_{k(+)}^{\mathrm{T}} \end{bmatrix} \mathrm{T}_{\mathrm{temp}} = \begin{bmatrix} \Theta & 0 \\ \Gamma & C_{Y_{k+1(-)}} \\ \tau^{\mathrm{T}} & \hat{s}_{k+1(-)}^{\mathrm{T}} \end{bmatrix}$$

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.

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

7.1 An $n \times n$ Moler matrix M has 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.

7.2 Write a MATLAB script to compute and print out the $n \times n$ Moler matrices and their lower triangular Cholesky factors for $2 \leq n \leq 20$.

7.3 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.

7.4 Show that if A and B are $n \times n$ upper triangular matrices, then their product AB is also upper triangular.

7.5 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?)

7.6 Show that the inverse of an upper (lower) triangular matrix is also an upper (lower) triangular matrix.

7.7 Show that if the upper triangular Cholesky decomposition algorithm is applied to the matrix product

$$\begin{bmatrix} H & z \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} H & z \end{bmatrix} = \begin{bmatrix} H^{\mathrm{T}}H & H^{\mathrm{T}}z \\ z^{\mathrm{T}}H & z^{\mathrm{T}}z \end{bmatrix}$$

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).

7.8 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 generalized Cholesky factor of P and a square root of P.

7.9 Show that the column vectors of the orthogonal matrix E in the singular-value decomposition of P (in the above exercise) 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 $Pe_i = d_i e_i$.

7.10 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.

7.11 Show that if C is an $n \times n$ generalized Cholesky factor of P, then, for any orthogonal matrix T, CT is also a generalized Cholesky factor of P.

7.12 Show that $(I - \nu\nu^{\mathrm{T}})^2 = (I - \nu\nu^{\mathrm{T}})$ if $|\nu|^2 = 1$ and that $(I - \nu\nu^{\mathrm{T}})^2 = I$ if $|\nu|^2 = 2$.

7.13 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 generalized Cholesky factors of R and $R + V^{\mathrm{T}}V$, respectively.

7.14 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[17]

$$\sum_{k=m}^{j} W_{ik} W_{mk} = \Delta_{im} - \frac{\mathbf{v}_i \mathbf{v}_m}{R + \sum_{k=1}^{j} \mathbf{v}_k^2} \tag{7.266}$$

for all i, m, j such that $1 \leq i \leq m \leq j \leq n$.

7.15 Prove that the Björck "modified" Gram–Schmidt algorithm results in a set of mutually orthogonal vectors.

7.16 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?

7.17 Show that if A and B are orthogonal matrices, then

$$\begin{bmatrix} A & 0 \\ 0 & B \end{bmatrix}$$

is an orthogonal matrix.

7.18 What is the inverse of the Householder reflection matrix $I - 2vv^{\mathrm{T}}/v^{\mathrm{T}}v$?

7.19 How many Householder transformations are necessary for triangularization of an $n \times q$ matrix when $n < q$? Does this change when $n = q$?

7.20 (Continuous temporal update of generalized Cholesky factors.) Show that all differentiable generalized 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 [44].

7.21 Prove that the condition number of the information matrix is equal to the condition number of the corresponding covariance matrix in the case that neither of them is singular. (The condition number is the ratio of the largest characteristic value to the smallest characteristic value.)

[17] Kronecker's delta (Δ_{ij}) is defined to equal 1 only if its subscripts are equal ($i = j$) and to equal zero otherwise.

7.22 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.)

7.23 Prove the correctness of the triangularization equation for the temporal update of the SRIF.

7.24 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.)

7.25 Show that, if C is a generalized Cholesky factor of P (i.e., $P = CC^{\mathrm{T}}$), then $C^{-\mathrm{T}} = (C^{-1})^{\mathrm{T}}$ is a generalized Cholesky factor of $Y = P^{-1}$, provided that the inverse of C exists. Conversely, the *transposed* inverse of any generalized Cholesky factor of the information matrix Y is a generalized Cholesky factor of the covariance matrix P, provided that the inverse exists.

7.26 Write a MATLAB script to implement Example 5.8 using the Bierman–Thornton UD filter, plotting as a function of time the resulting root-mean-square (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.)

7.27 Write a MATLAB script to implement Example 5.8 using the Potter "square-root" filter and plotting the same values as in the problem above.

REFERENCES

[1] L. Strachey, *Eminent Victorians*, Penguin Books, London, 1988.

[2] J. S. Meditch, "A survey of data smoothing for linear and nonlinear dynamic systems," *Automatica*, Vol. 9, pp. 151–162, 1973.

[3] S. R. McReynolds, "Fixed interval smoothing: revisited," *AIAA Journal of Guidance, Control, and Dynamics*, Vol. 13, pp. 913–921, 1990.

[4] G. J. Bierman, "A new computationally efficient fixed-interval discrete time smoother," *Automatica*, Vol. 19, pp. 503–561, 1983.

[5] 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.

[6] J. M. Jover and T. Kailath, "A parallel architecture for Kalman filter measurement update and parameter update," *Automatica*, Vol. 22, pp. 783–786, 1986.

[7] ANSI/IEEE Std. 754-1985, *IEEE Standard for Binary Floating-Point Arithmetic*, Institute of Electrical and Electronics Engineers, New York, 1985.

[8] 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 Veiw, CA, 1985.

[9] 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.

[10] R. H. Battin, "Space guidance evolution—a personal narrative," *AIAA Journal of Guidance and Control*, Vol. 5, pp. 97–110, 1982.

[11] 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.

[12] 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.

[13] J. E. Potter, "Matrix quadratic solutions," *SIAM Journal of Applied Mathematics*, Vol. 14, pp. 496–501, 1966.

[14] 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.

[15] J. J. Dongarra, C. B. Moler, J. R. Bunch, and G. W. Stewart, *LINPACK Users' Guide*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1979.

[16] G. E. Forsythe, M. A. Malcolm, and C. B. Moler, *Computer Methods for Mathematical Computations*, Prentice-Hall, Englewood Cliffs, NJ, 1977.

[17] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 4th ed., Johns Hopkins University Press, Baltimore, MD, 2013.

[18] P. G. Kaminski, Square root filtering and smoothing for discrete processes, PhD thesis, Stanford University, 1971.

[19] 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.

[20] 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.

[21] A. S. Householder, "Unitary triangularization of a nonsymmetric matrix," *Journal of the Association for Computing Machinery*, Vol. 5, pp. 339–342, 1958.

[22] N. A. Carlson, "Fast triangular formulation of the square root filter," *AIAA Journal*, Vol. 11, No. 9, pp. 1259–1265, 1973.

[23] W. S. Agee and R. H. Turner, *Triangular Decomposition of a Positive Definite Matrix Plus a Symmetric Dyad, with Applications to Kalman Filtering*, Technical Report No. 38, White Sands Missile Range Oct. 1972.

[24] G. J. Bierman, *Factorization Methods for Discrete Sequential Estimation*, Academic Press, New York, 1977.

[25] C. L. Thornton, Triangular covariance factorizations for Kalman filtering, PhD thesis, University of California at Los Angeles, 1976.

[26] Å. Björck, "Solving least squares problems by orthogonalization," *BIT*, Vol. 7, pp. 1–21, 1967.

[27] M. S. Grewal and J. Kain, "Kalman filter implementation with improved numerical properties," *IEEE Transactions on Automatic Control,* Vol. 55, No. 9, pp. 2058–2068, 2010.

[28] 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.

[29] L. F. Richardson, "The approximate arithmetical solution by finite differences of physical problems including differential equations, with an application to the stresses in a masonry dam," *Philosophical Transactions of the Royal Society A*, Vol. 210, No. 459–470, pp. 307–357, 1911.

[30] J. Stoer and R. Bulirsch, *Introduction to Numerical Analysis*, 3rd ed., Springer-Verlag, New York, 2002.

[31] C. Moler and C. Van Loan, "Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later," *SIAM Review*, Vol. 45, No. 1, pp. 3–49, 2003.

[32] C. F. Van Loan, "Computing integrals involving the matrix exponential," *IEEE Transactions on Automatic Control*, Vol. AC-23, pp. 395–404, 1978.

[33] 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.

[34] 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.

[35] V. Strassen, "Gaussian elimination is not optimal," *Numerische Matematik*, Vol. 13, p. 354, 1969.

[36] R. E. Kalman, "A new approach to linear filtering and prediction problems," *ASME Journal of Basic Engineering*, Vol. 82, pp. 34–45, 1960.

[37] R. H. Battin, *Astronautical Guidance*, McGraw-Hill, New York, 1964.

[38] 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, pp. 333-1–333-13, 1963.

[39] R. S. Bucy and P. D. Joseph, *Filtering for Stochastic Processes with Applications to Guidance*, American Mathematical Society, Chelsea Publishing, Providence, RI, 2005.

[40] M. Morf and T. Kailath, "Square root algorithms for least squares estimation," *IEEE Transactions on Automatic Control*, Vol. AC-20, pp. 487–497, 1975.

[41] G. H. Golub, "Numerical methods for solving linear least squares problems," *Numerische Mathematik*, Vol. 7, pp. 206–216, 1965.

[42] C. L. Lawson and R. J. Hanson, *Solving Least Squares Problems*, Prentice-Hall, Englewood Cliffs, NJ, 1974.

[43] 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.

[44] A. Andrews, "A square root formulation of the Kalman covariance equations," *AIAA Journal*, Vol. 6, pp. 1165–1166, 1968.

8

NONLINEAR APPROXIMATIONS

I suppose it is tempting, if the only tool you have is a hammer, to treat everything as if it were a nail

—Abraham H. Maslow
The Psychology of Science: A Reconnaissance, Harper, 1966.

8.1 CHAPTER FOCUS

Although the Kalman filter (our hammer) was developed to nail the *linear* least-mean-squares estimation problem, it has since been applied with impunity—and considerable success—to a variety of important less-than-linear problems. As was mentioned in Chapter 3, the introduction of nonlinearity in the dynamic or measurement model corrupts the standard (linear) propagation of the mean and covariance by coupling *unknown* higher order moments into the mean and covariance.

8.1.1 Topics to Be Covered

Methods proposed for coping with such nonlinearities generally fall into four broad categories:

I. Extension to affine dynamic and measurement models, which are equivalent to the standard Kalman filter model with nonzero-mean dynamic disturbances and/or measurement noise. The solution is trivial and exact.

Kalman Filtering: Theory and Practice Using MATLAB®, Fourth Edition.
Mohinder S. Grewal and Angus P. Andrews.
© 2015 John Wiley & Sons, Inc. Published 2015 by John Wiley & Sons, Inc.

II. Linear approximation of

A. The variation of measurement variables with changes in state variables, usually by partial derivatives which may be either

1. Analytical, by taking partial derivatives of differentiable nonlinear functions $z = h(x)$, or
2. Numerical, by evaluating $h(x)$ at the estimated state $h(\hat{x})$ and at "perturbations" $h(\hat{x} + \delta_j)$, and computing the ratios

$$\left.\frac{\partial z_i}{\partial x_j}\right|_{x=\hat{x}} \approx \frac{h_i(\hat{x} + \delta_j) - h_i(\hat{x})}{\{\delta_j\}_j},$$

where $h_i(\cdot)$ is the ith component of h and the components of δ_j are nonzero only in the jth row. Then the linearized approximation of the measurement sensitivity matrix H will have the numerical approximation in the ith row and jth column.

In either case, these approximations of H are only used for computing the Kalman gain. The expected measurement is always computed as

$$\hat{z} = h(\hat{x}).$$

B. Covariance dynamics, again by using either analytical or numerical partial differentiation for the purpose of propagating the covariance matrix of state estimation uncertainty by any of the four possible methods:

1. If the nonlinear dynamic model has the form

$$\dot{x} = f(x),$$

use the analytical partial derivative

$$F = \frac{\partial f}{\partial x}$$

to propagate P using

$$\dot{P} = FP + PF^{\mathrm{T}} + Q(t).$$

2. Use the value of F from (II.B.1) to compute

$$\Phi_{k-1} \approx \exp\left(\int_{t_{k-1}}^{t_k} F(s)\, ds\right).$$

3. Approximate Φ by propagating $\dot{x} = f(x)$ from t_{k-1} to t_k for $n+1$ different initial values of x, including $\hat{x}_{(k-1)(+)}$ and perturbations δ_j along each of the n state vector components. Then

$$\{\Phi_{k-1}\}_{ij} \approx \frac{\{x_{kj} - \hat{x}_{k(-)}\}_i}{\delta_j},$$

where x_{kj} is the perturbed solution at time t_k for the trajectory with initial value $\hat{x}_{(k-1)(+)} + \delta_j$.

4. If the nonlinear model has the form

$$x_k = f_k(x_{k-1}),$$

then the linear approximation of the state-transition matrix is the Jacobian matrix

$$\Phi_{k-1} \approx \frac{\partial f_k}{\partial x_{k-1}}.$$

In all cases, these approximations are only for propagating the covariance matrix P, not the estimate $\hat{x}$. The full nonlinear model is always used for propagating $\hat{x}$.

III. Replacing the matrix Riccati differential or difference equation with statistical approximations using discrete samples. These "sample-and-propagate" methods are similar in some ways to (II.B.3), except that they make more intelligent use of perturbations, and can carry the covariance solutions all the way through the measurements. They perform better than linearization methods for some applications—but they, too, have their limitations.

IV. Deriving something roughly equivalent to the Kalman filter, except with some limited model nonlinearities. These have included

A. A derivation including terms out to second order by Bass *et al.* [1].

B. A derivation including terms out to third order by Wiberg and Campbell [2].

C. A derivation by Beneš [3] with limited nonlinearities, but with a closed-form solution.

The computational complexities of the second- and third-order derivations are generally too extreme to justify their limited advantages, and they are only mentioned here. The Beneš filter will be described here, although it has not performed well against filters from Categories II or III in some studies [29].

Beyond approaches to modifying the Kalman filter implementation to better cope with model nonlinearities, there has been a long history of nonlinear stochastic system modeling. Although it has perhaps not produced anything more "snappy" than

the Kalman filter, it has had some notable successes in modeling the behavior of thermodynamic systems, economic systems, and market systems—among others. It has also produced some "grid-based" implementations that have been used for computing probability densities for surveillance (detection and tracking) problems.

The approximation approaches generally depend on the fact that the Kalman filter is optimal for its model of the problem. Although some might expect optimal solutions of this sort to be brittle in some way, it is generally not the case. Optimizing solutions with respect to quadratic criteria such as mean-squared error tends to make derivatives of their first-order performance with respect to model variations zero, so that minor modifications of the model can have acceptable influences on optimality.

This chapter is about some of the more successful approaches.

8.1.2 What Does "Nonlinear" Mean?

It has been said that *"Classification of mathematical problems as linear and nonlinear is like classification of the Universe as bananas and non-bananas,"* the point being that the vast majority of real-world mathematical problems are not linear in nature.

We then need to qualify what we mean by "nonlinear," which is perhaps too strong a term for the more successful nonlinear applications of Kalman filtering. There are just too many wild things parading as mathematical functions to include them all, and we have shown in Chapter 3 what happens to the propagation of the fundamental variables of Kalman filtering (means and covariances) when some rather bland nonlinearities are introduced.

Many of the so-called "nonlinear" problems on which these methods have been successful are perhaps better described as being "quasilinear," in that the nonlinear functions involved are locally dominated their first-order variations—at least within the variations expected by the probability distributions of the estimation errors and measurement errors. As a consequence, the errors introduced by linear approximation depend on the quality of the estimates and measurements, and the success of a "nonlinear" application of Kalman filtering depends only on the linearization error within ranges determined by estimation uncertainty.

8.2 THE AFFINE KALMAN FILTER

This is a trivial extension of the linear Kalman filter to what might be called an *affine Kalman filter.* As we have seen in Chapter 3, affine transformations are not exactly linear, but their effects on probability distributions are not much different from those of linear transformations—especially their effects on the means and covariances, which are not corrupted by other moments of the distribution. We mention it here because it is also equivalent to the standard linear Kalman filter model, except with nonzero-mean noise sources.

The Kalman filter with control inputs is essentially an affine Kalman filter.

8.2.1 Affine Model

Affine transformations are linear transformations with an added known and constant "bias,"

$$x_k = \Phi_{k-1}x_{k-1} + b_{k-1} + w_{k-1} \tag{8.1}$$

$$z_k = H_k x_k + d_k + v_k, \tag{8.2}$$

where b_{k-1} is a known n-vector, d_k is a known m-vector, and both $\{w_{k-1}\}$ and $\{v_k\}$ are zero-mean white noise sequences with known covariances Q_{k-1} and R_k, respectively.

8.2.2 Nonzero-Mean Noise Model

The affine model is indistinguishable from the nonzero-mean noise model in which

$$\underset{w}{\mathrm{E}}\langle w_{k-1}\rangle = b_{k-1} \tag{8.3}$$

$$\underset{w}{\mathrm{E}}\langle (w_{k-1} - b_{k-1})(w_{k-1} - b_{k-1})^{\mathrm{T}}\rangle = Q_{k-1} \tag{8.4}$$

$$\underset{v}{\mathrm{E}}\langle v_k\rangle = d_k \tag{8.5}$$

$$\underset{v}{\mathrm{E}}\langle (v_k - d_k)(v_k - d_k)^{\mathrm{T}}\rangle = R_k. \tag{8.6}$$

The extension of the Kalman filter to include nonzero-mean noise is therefore equivalent to the affine extension.

8.2.3 Affine Filter Implementation

The equivalent implementation of the affine Kalman filter is then

$$\hat{x}_{k(-)} = \Phi_{k-1}\hat{x}_{(k-1)(+)} + b_{k-1} \tag{8.7}$$

$$P_{k(-)} = \Phi_{k-1}P_{(k-1)(+)}\Phi_{k-1}^{\mathrm{T}} + Q_{k-1}$$

$$\hat{z}_k = H_k\hat{x}_{k(-)} + d_k \tag{8.8}$$

$$\overline{K}_k = P_{k(-)}H_k^{\mathrm{T}}[H_kP_{k(-)}H_k^{\mathrm{T}} + R_k]^{-1}$$

$$\hat{x}_{k(+)} = \hat{x}_{k(-)} + \overline{K}_k(z_k - \hat{z}_k) \tag{8.9}$$

$$P_{k(+)} = P_{k(-)} - \overline{K}_kH_kP_{k(-)},$$

which differs from the linear Kalman filter only in the numbered equations.

8.3 LINEAR APPROXIMATIONS OF NONLINEAR MODELS

8.3.1 Linearizing the Riccati Differential Equation

Given an estimation problem with possibly nonlinear state dynamic model and measurement model with zero-mean error sources

$$\dot{x} = f(x,t) + w(t) \tag{8.10}$$

$$Q(t) \overset{\text{def}}{=} \underset{w}{\mathrm{E}}\langle w(t)w^{\mathrm{T}}t\rangle$$

$$z_k = h_k(x(t_k)) + v_k \tag{8.11}$$

$$R_k \overset{\text{def}}{=} \underset{v}{\mathrm{E}}\langle v_k v_k^{\mathrm{T}}\rangle,$$

the state estimate $\hat{x}$ can be propagated between measurement epochs by integrating

$$\frac{d}{dt}\hat{x}(t) = f(\hat{x}(t),t) \tag{8.12}$$

and the predicted measurement can be computed as

$$\hat{z}_k = h_k(\hat{x}_{k(-)}). \tag{8.13}$$

However, there is no comparable general solution for nonlinear propagatiion of the covariance matrix P of estimation uncertainty—which is needed for computing the Kalman gain.

If the vector-valued functions $f(\cdot,t)$ and $h_k(\cdot)$ are sufficiently differentiable, then their Jacobian matrices

$$F(x,t) \overset{\text{def}}{=} \frac{\partial f(x,t)}{\partial x} \tag{8.14}$$

$$H_k(x) \overset{\text{def}}{=} \frac{\partial h_k(x)}{\partial x} \tag{8.15}$$

can be used as a linear approximation for propagating $\hat{x}$ and P as the solution to

$$\dot{P}(t) \approx F(\hat{x}(t),t)P(t) + P(t)F^{\mathrm{T}}(\hat{x}(t),t) + Q(t) \tag{8.16}$$

$$\overline{K}_k = P_{k(-)}H_k^{\mathrm{T}}(x)[H_k(x)P_{k(-)}H_k^{\mathrm{T}}(x) + R_k]^{-1} \tag{8.17}$$

$$\hat{x}_{k(+)} = \overline{K}_k[z_k - h(\hat{x}_{k(-)},t_k)] \tag{8.18}$$

$$P_{k(+)} = P_{k(-)} - \overline{K}_k P_{k(-)}H_k(x). \tag{8.19}$$

This approach only works if the linear approximations are "close enough for all practical purposes"—a concept that will be made more rigorous in Section 8.3.4.

Equation 8.14 assumes that $f(x,t)$ is a differentiable function of x. If its derivative (Jacobean matrix) is a known matrix function of x and t, then that derivative function is $F(x,t)$.

If the function $f(x,t)$ of Equation 8.10 were linear, then it could be expressed as

$$f(x,t) = F(t)x \tag{8.20}$$

$$F(t) = \frac{\partial f(x,t)}{\partial x}. \tag{8.21}$$

Otherwise, the partial derivative with respect to x will still be a function of x. In that case, it is necessary to assume some value of x ($\hat{x}$ will do) in evaluating the partial derivative.

8.3.2 Approximating Φ with Numerical Partial Derivatives

Even if $f(x,t)$ is a differentiable function of x, it is sometimes more efficient to propagate the covariance matrix using a state-transition matrix Φ approximated by numerical partial differentiation. This approach was used in early analysis and implementation of space navigation and control, in which the dynamic equations are for free-fall under the gravitational influence of multiple massive bodies. The same algorithms developed to generate the trajectory of the estimate $\hat{x}$ could then be used to generate "nearby" trajectories defined by perturbations from the initial conditions of $\hat{x}$. If we let the perturbed initial conditions be

$$x_{(k-1)\,[\ell]} = \hat{x}_{k-1} + \delta_{\,[\ell]}, \tag{8.22}$$

where $\delta_{\,[\ell]}$ has zeros except in its ℓth component, then the respective trajectory solutions at time t_k can be used to approximate the state-transition matrix as

$$\Phi_k \approx \frac{\partial x_k}{\partial x_{k-1}} \tag{8.23}$$

$$\phi_{kij} \approx \frac{x_{(k-1)(-)\,[j]i} - \hat{x}_{k(-)i}}{x_{(k-1)(+)\,[j]\,j} - \hat{x}_{(k-1)(+)j}}. \tag{8.24}$$

Many of these perturbation techniques had already been developed before the Space Age, during which they were used for satellite tracking, satellite navigation, and targeting and guiding ballistic missiles. Major players in this effort included William H. Guier (1926–2011) and George C. Weiffenbach (1921–2003) at the Applied Physics Laboratory of Johns Hopkins University and J. Halcombe Laning (1920–2012) and Richard H. Battin at the MIT Instrumentation Laboratory.

8.3.3 Linearized and Extended Kalman Filters

8.3.3.1 Linearized Kalman Filtering The Apollo Project to send Americans to the surface of the moon and back was announced by President John Fitzgerald Kennedy before a joint session of Congress on May 25, 1961. At that time, Stanley F. Schmidt was already studying the associated navigation and guidance problem at the NASA Ames Research Center in Mountain View, California. It was already a standard practice to use numerical partial differentiation to linearize problems, and Schmidt had been the first to recognize the potential of Kalman filtering for this application. For that purpose, he had been directing feasibility studies using perturbations evaluated about a "nominal" earth-to-moon-and-return trajectory. That is, the partial derivative approximations

$$\Phi_k \approx \left.\frac{\partial x_k}{\partial x_{k-1}}\right|_{x=x_N} \tag{8.25}$$

$$H_k \approx \left.\frac{\partial h_k(x)}{\partial x}\right|_{x=x_N}, \tag{8.26}$$

where x_N denotes a nominal trajectory.

This approach is quite common in preliminary design studies, when the actual trajectories are not known precisely. With this technique, performance requirements for the Apollo navigation sensors (a "space sextant" mounted on an inertial platform) could be determined by varying the associated error covariances to determine the minimum sensor performance required for mission success.

These preliminary performance studies need not use the full Kalman filter, but only the covariance computations of the Riccati equation. Linearized Kalman filter models are used almost exclusively for solving Riccati equations, not for state estimation.

Covariance analysis using linearized models is also used for assessing the expected level of linearization errors (Section 8.3.4) from extended Kalman filtering.

8.3.3.2 Extended Kalman Filtering Stanley F. Schmidt was the first to propose that the linearization technique could be adapted for onboard navigation *by using partial derivatives evaluated at the current time and estimated state variable.* That is,

$$\Phi_k \approx \left.\frac{\partial x_k}{\partial x_{k-1}}\right|_{x=\hat{x}} \tag{8.27}$$

$$H_k(\hat{x}) \approx \left.\frac{\partial h_k(x)}{\partial x}\right|_{x=\hat{x}}, \tag{8.28}$$

where $\hat{x}$ denotes the estimated trajectory.

Schmidt called this the *extended Kalman filter* (EKF). Others at the time called it the *Kalman–Schmidt filter*.[1]

There are many ways in which the linear approximations of the linearized and EKFs can be implemented, however.

[1] A later innovation of Schmidt's would come to be called the *Schmidt–Kalman filter* (SKF).

Example 8.1 (Analytical Linear Approximation EKF for n–body Dynamics) For more than half a century, the navigation, tracking, and control of spacecraft has made use of linearized and/or extended Kalman filtering models for the gravitational dynamics of travel in "free space," where the gravitational acceleration of a body of relatively insignificant mass m located at X due to a body of significant mass M located at Y some distance R from X is given by Newton's third law as

$$\begin{aligned}\ddot{X} &= g(X)\\ &= \frac{GM(Y-X)}{R^3}\\ R &= [(Y_1-X_1)^2+(Y_2-X_2)^2+(Y_3-X_3)^2]^{1/2},\end{aligned}$$

Where G is the gravitational constant and X and Y are specified as inertial (i.e., nonrotating and nonaccelerating) coordinates.

The gravity gradient matrix for a single massive body is then expressible as an analytical partial derivative,

$$\frac{\partial \ddot{X}}{\partial X} = -\frac{GM}{R^3} I_3 - \frac{3GM}{R^5}(Y-X)(Y-X)^{\mathrm{T}}.$$

Because accelerations are additive, the net acceleration on the small body at X from n such massive bodies of masses M_i and located at Y_i is the sum,

$$\begin{aligned}\ddot{X} &= G\left[\sum_{i=1}^{N} \frac{M_i\left(Y_i - X\right)}{R_i^3}\right]\\ R_i &= |Y_i - X|,\end{aligned} \tag{8.29}$$

and the net gravity gradient

$$\frac{\partial \ddot{X}}{\partial X} = -G\left[\sum_{i=1}^{n} \frac{M_i}{R_i^3}\right] I_3 - 3G\left[\sum_{i=1}^{n} \frac{M_i}{R_i^5}\left(Y_i - X\right)\left(Y_i - X\right)^{\mathrm{T}}\right]. \tag{8.30}$$

In this dynamic model, the vectors Y_i are known functions of time, and the influence of the mass m on their trajectories is considered to be insignificant.

One can then write a nonlinear state-space model for the dynamics of Equation 8.29 in terms of a 6-vector as

$$\begin{aligned}x(t) &\stackrel{\text{def}}{=} \begin{bmatrix} X(t) \\ \dot{X}(t) \end{bmatrix}\\ \dot{x}(t) &= f(x,t)\\ &= \begin{bmatrix} \dot{X}(t) \\ \ddot{X}(t) \end{bmatrix},\end{aligned}$$

and any initial value $x(t_0)$ can be propagated forward in time using Equations 8.29.

In the analytical EKF approximation, propagation of the covariance matrix P of x can use a first-order dynamic coefficient matrix approximated by partial differentiation of f with respect to x:

$$F(t) \stackrel{\text{def}}{=} \left.\frac{\partial f}{\partial x}\right|_{\hat{x}} = \begin{bmatrix} 0 & I_3 \\ \left.\frac{\partial \ddot{X}}{\partial X}\right|_{\hat{x}(t)} & 0 \end{bmatrix},$$

which can be evaluated using Equation 8.30.

The resulting formula can be used to propagate the covariance matrix P in a couple of ways:[2]

$$\begin{aligned} \dot{P}(t) &= F(t)P(t) + P(t)F^{\mathrm{T}}(t) + Q(t), \text{ or} \\ P_{k(-)} &= \Phi_{k-1}P_{k(-)}\Phi_{k-1}^{\mathrm{T}} + Q_k \\ \Phi_{k-1} &\stackrel{\text{def}}{=} \exp\left[\int_{t_{k-1}}^{t_k} F(s)\, ds\right]. \end{aligned}$$

The EKF process flow using the first of these methods is diagrammed in Figure 8.1, where analytical partial derivatives are also used for approximating the measurement sensitivity matrix H_k.

Note, however, that the predicted state vector $\hat{x}_{k(-)}$ and measurement vector $\hat{z}_k$ are computed without linear approximation. That is the characteristic of all EKF implementations.

Example 8.2 (Numerical Linear Approximation EKF for n-body Dynamics) There is yet a third way to implement an EKF for the same nonlinear dynamics problem as Example 8.1, this time using numerical partial derivatives to estimate the state-transition matrix Φ.

In this implementation, Equation 8.29 is not only used for propagating the estimate $\hat{x}_{(k-1)(+)}$ to obtain $\hat{x}_{k(-)}$, but also for propagating six perturbed initial values

$$\begin{aligned} S_j(t_{k-1}) &= \hat{x}_{(k-1)(+)} + \delta_j \\ \{\delta_j\}_i &= \begin{cases} \varepsilon_j \neq 0, & i = j \\ 0, & i \neq j \end{cases}, \end{aligned}$$

where $\{\delta_j\}_i$ denotes the ith component of δ_j.

If the trajectory with initial value $S_j(t_{k-1})$ at time t_{k-1} has solution

$$S_j(t_k) = \hat{x}_{k(-)} + \Delta_j$$

[2]These models show dynamic disturbance noise covariance $Q(t)$ or Q_k, although dynamic disturbance noise is insignificant for space trajectories.

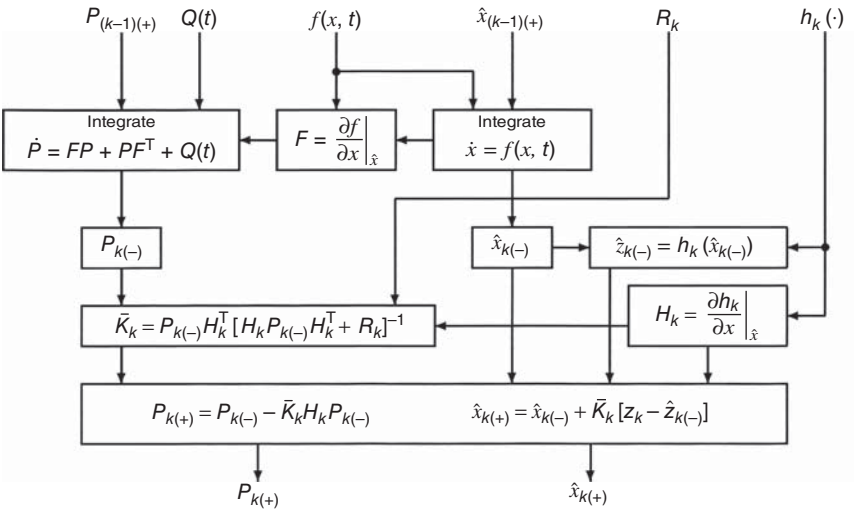


Figure 8.1 EKF process flow using analytical partial derivatives.

at time t_k, then the numerical partial derivative approximation of Φ_{k-1} will have the value

$$\phi_{ij} \approx \frac{\{\Delta_j\}_i}{\varepsilon_j}$$

in its ith row and jth column, where $\{\Delta_j\}_i$ is the ith component of the vector Δ_j.

The EKF process flow using this approach is diagrammed in Figure 8.2. This resembles, in some ways, that of the unscented Kalman filter (UKF) diagrammed in Figure 8.19.

Example 8.3 (Analytical Linear Approximation for the 2–body Problem) Not all linearized or EKF applications require integrating the differential equations of a dynamic model for propagating the trajectory solution forward in time. In those cases for which a closed-form analytical solution already exists, an approximation of the state-transition matrix Φ can be obtained by taking analytical partial derivatives of the closed-form "initial value" solution with respect to initial conditions.

In the case of the two-body problem, such as that for an orbit about the sun—neglecting the minor influences of the other planets—the solution is provided by Kepler's equations, which can be manipulated to yield the position and velocity trajectory as a function of time and initial conditions. The result is a set of formulas for $x(t_k)$ as a function of $x(t_{k-1})$, which is sufficient for propagating the estimate. It also provides an approximation for the linearized state-transition matrix Φ_{k-1} as the Jacobian matrix

$$\Phi_{k-1} \approx \left.\frac{\partial x\left(t_k\right)}{\partial x(t_{k-1})}\right|_{\hat{x}_{(k-1)(+)}}.$$

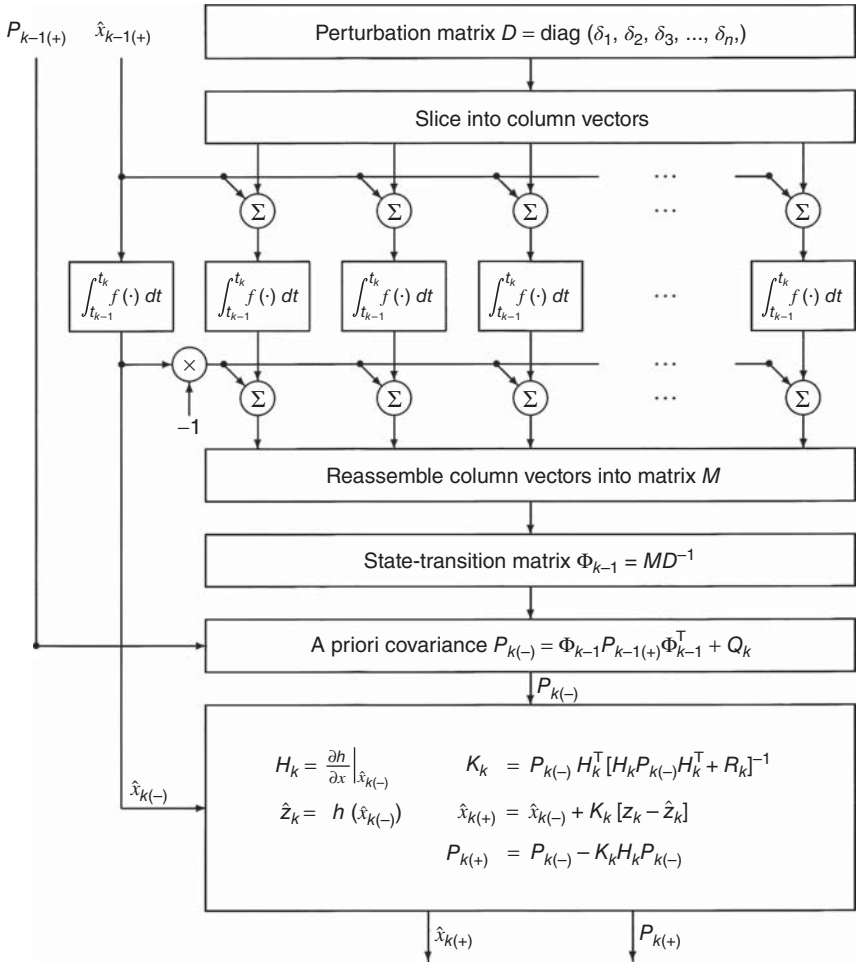


Figure 8.2 EKF data flow using numerical differentiation of $\dot{x} = f(x, t)$.

For the case with $n = 2$ bodies, this provides a fourth alternative EKF solution for the same problem as Examples 8.1 and 8.2. However, this closed-form solution does not provide adequately accurate EKF estimates for high precision applications such as estimating the emphemerides of Global Navigation Satellite System (GNSS) satellites or ICBM guidance. These applications cannot ignore the nonspherical anomalies of Earth's gravitational field or the gravitational effects of the sun and moon.

Example 8.4 (Dynamic Model Parameter Estimation) Estimating the parameters of a Kalman filtering model is a notoriously nonlinear problem, but a fairly common one. This example uses the linear damped harmonic oscillator model from Example 5.7, and as a nonlinear model parameter estimation problem, assuming that

ζ (damping coefficient) is an unknown constant. Therefore, the damping coefficient can be modeled as a state variable and its value estimated using an EKF.

Let

$$x_3(t) = \zeta$$

and

$$\dot{x}_3(t) = 0.$$

Then the otherwise linear dynamic equation in continuous time becomes nonlinear:

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \dot{x}_3(t) \end{bmatrix} = \begin{bmatrix} x_2 \\ -\omega^2 x_1 - 2x_2 x_3 \quad \omega \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} w(t) + \begin{bmatrix} 0 \\ 12 \\ 0 \end{bmatrix}.$$

The observation equation is still linear, however:

$$z(t) = x_1(t) + v(t)$$

One hundred data points were simulated with random plant noise and measurement noise, $\zeta = 0.1, \omega = 10$ rad/s, and initial conditions

$$\begin{bmatrix} x^1(0) \\ x^2(0) \\ x^3(0) \end{bmatrix} = \begin{bmatrix} 0 \text{ ft} \\ 0 \text{ ft/s} \\ 0 \end{bmatrix}, \qquad P(0) = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix},$$

$$Q = 4.47(\text{ft/s})^2, \qquad R = 0.001(\text{ft}^2).$$

The discrete nonlinear plant and linear observation equations for this model are

$$\begin{aligned} x_k^1 &= x_{(k-1)}^1 + T x_{(k-1)}^2 \\ x_k^2 &= -25T x_{(k-1)}^1 + (1 - 10T x_{(k-1)}^3) x_{(k-1)}^2 + 12T + T w_{k-1} \\ x_k^3 &= x_{(k-1)}^3 \\ z_k &= x_k^1 + v_k. \end{aligned}$$

Figure 8.3 shows the estimated position, velocity, and damping factor states using an EKF. This is implemented in MATLAB®script `DampParamEst.m`, which runs successive calls with independent random samples. (Because this is a Monte Carlo simulation, successive calls will not produce the same results.)

Example 8.5 (Nonlinear Freeway Traffic Modeling with EKF) This is an application of extended Kalman filtering to estimating parameters of an already nonlinear dynamic model. The objective of this particular task was to estimate certain key parameters of a macroscopic freeway traffic model designed to simplify a driver-level "microscopic" model [4].

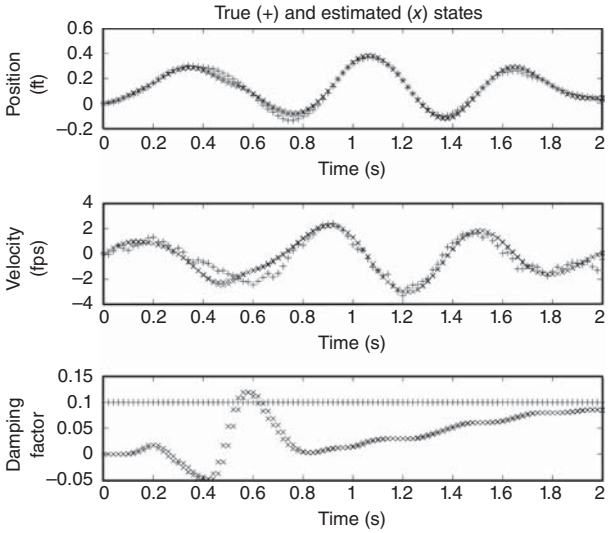


Figure 8.3 EKF estimation of oscillator position, velocity, and damping factor.

Macroscopic-Level Freeway Traffic Modeling: This particular application was designed to represent freeway traffic dynamics in terms of the speed and traffic densities within a 42-mile-long freeway circuit through Los Angeles and Long Beach, California. This was an early application of modern estimation and control methods to the problems of improving performance of urban infrastructure. Traffic on freeways had increased to the point that their carrying capacity was diminished, and something had to be done to prevent gridlock. Controlling the rate of access at on-ramps was the preferred method, but it had to be coupled with a reasonably accurate dynamic model for determining how access control influences traffic dynamics, and the model had to represent how real drivers respond to changing traffic conditions.

"Microscopic" vehicle-level simulations with driver response models developed for offline traffic simulations were good at representing actual freeway traffic dynamics, but were not fast enough for real-time controller-in-the-loop implementation. What was needed was a system-level "macroscopic" dynamic model that could be abstracted from vehicle-level dynamical behavior and used as part of real-time traffic control. The resulting model includes parameters that would need to be tuned to agree with results of the microscopic offline vehicle-level simulations.

For modeling purposes, a representative freeway circuit was divided into a sequence of contiguous segments, as illustrated in Figure 8.4.[3] Traffic dynamics along a stretch of freeway with N segments require N such state variables, x_j; $j = 1, 2, 3, \cdots, N$, whereas the microscopic model might have thousands of

[3]The segments shown in the figure are short, for illustrative purposes only. Segment lengths were more typically in the order of a mile or less.

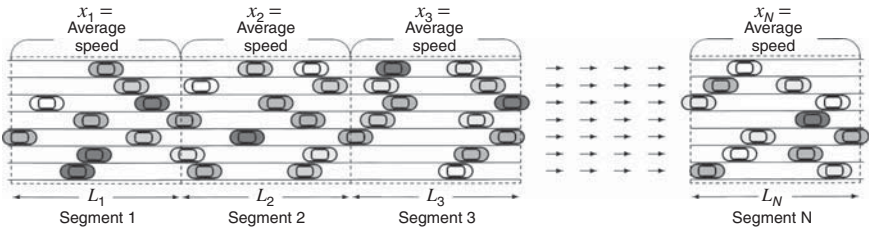


Figure 8.4 Freeway traffic model geometry (not to scale).

simulated vehicles. These segments may also have different lengths, modeled by the parameters L_j representing the lengths of the numbered segments. Generally, segment lengths are chosen so that driver responses are determined by conditions within a segment and in the segment ahead.

Nonlinear Dynamic Model: The core state variables are the average traffic speeds within the segments, as illustrated in the figure. The traffic dynamic model included driver responses to local conditions (speed, density, and their longitudinal gradients) within the various segments, including small random excursions due to lags in driver perceptions and actions.

The resulting discrete-time traffic dynamic model has the abstract form

$$x_k = \phi(x_{k-1}, p_1, \ \dots \ , p_4) + w_{k-1}, \quad w_{k-1} \in \mathcal{N}(0, Q),$$

where the p_i are four unknown parameters and w_{k-1} is a zero-mean Gaussian white noise sequence with known covariance Q. The jth component of the dynamic function ϕ is

$$\begin{aligned}
&\phi_j(x_{k-1}, p_1, \ \dots \ , p_4) \\
&\quad \stackrel{\text{def}}{=} x_{(k-1),\,j} + \Delta t \left\{ -x_{(k-1),j} \frac{x_{(k-1),j} - x_{(k-1)(j-1)}}{L_j} \text{(speed gradient)} \right. \\
&\qquad - p_1 [x_{(k-1),\,j} - \underbrace{(p_2 + p_3 \rho_{(k-1),j})}_{\text{(equilibrium speed)}}] \\
&\qquad \left. + p_4 \left[\frac{(\rho_{(k-1)(j+1)} - \rho_{(k-1),j})}{\rho_{(k-1),j} L_j} \right] \right\} \text{(density gradient)},
\end{aligned}$$

where the top line of the model equation includes effects due to longitudinal speed gradients, the second line (the one with p_1) includes an observed effect of density on speed, and the last line includes effects due to longitudinal density gradients. The various variables and parameters of the model are

$\Delta t \stackrel{\text{def}}{=}$ discrete time interval,

$L_j \stackrel{\text{def}}{=}$ jth freeway segment length (miles),

$\rho_{(k-1),j} \stackrel{\text{def}}{=}$ traffic density in jth segment (vehicles per mile),

$p_1 \stackrel{\text{def}}{=}$ reciprocal of driver reaction time constant (1/s), an unknown parameter,

$p_2 \stackrel{\text{def}}{=}$ equilibrium traffic speed at zero density, an unknown parameter,

(should be somewhere near the speed limit),

$p_3 \stackrel{\text{def}}{=}$ equilibrium speed sensitivity to density, an unknown parameter,

$p_4 \stackrel{\text{def}}{=}$ acceleration response to density gradient, an unknown parameter.

Measurement Model: Observations include the average speeds of vehicles within each segment, as simulated by the microscopic model:

$$z_k = x_k + v_k, v_k \in \mathcal{N}(0, R),$$

where v_k is a zero-mean Gaussian white noise sequence with known covariance R.

Performance Metrics: The primary measures of performance for control purposes are the throughputs of the segments, defined by the numbers of vehicles per unit time leaving the various segments between sample times t_{k-1} and t_k. The objective of this initial study was to develop reliable estimation methods for the parameters and to test the accuracy of a model with the resulting parameters in predicting traffic conditions across traffic segments.

Least-Squares Parameter Estimation: The parameters p_2 and p_3 form a linear model of mean speed as a function of traffic density. Their values could be estimated by straightforward least-squares curve filtering to speed and density data from the microscopic model simulations, as shown in Figure 8.5. This is a plot of simulated vehicle speeds at different traffic densities, which shows a definite trend of mean speed as a function of traffic density.

Parameter Estimation with Extended Kalman Filtering: The remaining unknown parameters are p_1 and p_4, which are appended as augmented state variables:

$$x_{(N+1)} \stackrel{\text{def}}{=} p_1$$

$$x_{(N+2)} \stackrel{\text{def}}{=} p_4,$$

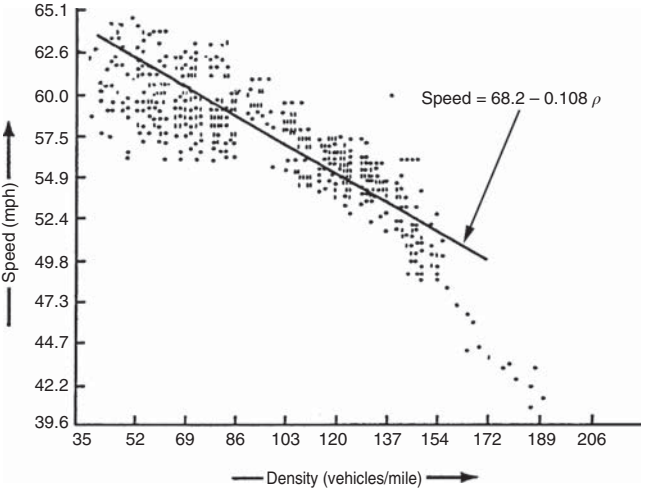


Figure 8.5 Speed–density relationship.

so that the augmented state vector $x^\star$ becomes

$$x^\star = \begin{bmatrix} x \\ x_{N+1} \\ x_{N+2} \end{bmatrix}, \quad x \overset{\text{def}}{=} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_N \end{bmatrix},$$

EKF Dynamic Model: The augmented dynamic system model is then linearized for EKF implementation as

$$x_k^\star = \Phi_{k-1} x_{k-1}^\star + \begin{bmatrix} w_{k-1} \\ 0 \\ 0 \end{bmatrix}$$

$$\Phi_{k-1} \overset{\text{def}}{=} \left.\frac{\partial \phi}{\partial x^\star}\right|_{x^\star = \hat{x}_{k-1}^\star}$$

$$= \left[\begin{array}{c|c} \left.\frac{\partial \phi}{\partial x}\right|_{x=\hat{x}_{k-1}} & \left.\frac{\partial \phi}{\partial x_{N+1}, x_{N+2}}\right|_{x^\star=\hat{x}_{k-1}^\star} \\ \hline 0 & I_2 \end{array}\right].$$

The augmented observation equation then becomes

$$z_k = H_k^\star x_k^\star + v_k$$
$$H_k^\star = \left[I_{N\times N} \mid 0_{N\times 2}\right].$$

Implementation: A four-lane, 6000-ft-long section of freeway with no on- or off-ramps and no accidents was simulated on a mainframe computer using a microscopic model. Eight files of data sets (high flow cases), each containing 20 min of data, along with segment mean speeds and densities at 1.5-s intervals were collected.

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 is obtained from a least-squares straight-line fit to the above data, as shown in Figure 8.5. The parameters p_2 and p_3 were estimated by this procedure.

As a rule, the choice of time-step size tends to be driven by the bandwidth of the dynamic model, and the choice of segment length is chosen to isolate dynamic interactions to adjacent segments only. For this application, a time-step size $\Delta t = 4.5$ s and the segment length $L_j = 0.5$ miles were chosen for stability considerations.

Parameter Estimation Results: An EKF was used to estimate p_1 and p_4 from the above data. For purposes of numerical computation, it is convenient to define dimensionless variables through the use of nominal values. The initial values used in the parameter identification algorithm are as follows:[4]

$$\text{Nominal segment mean speed} = 40 \text{ mph},$$

$$\text{Initial value of estimated driver reaction time } 1/p_1 = 30 \text{ s},$$

$$\text{Initial value of estimated density gradient sensitivity factor } p_1 p_4 = 4.0 \text{ mi}^2/\text{h}.$$

Figures 8.6 and 8.7 show convergence of the estimates of p_1 and p_4 during several minutes of estimation.

Performance Results: To test the resultant model as a predictor of future traffic conditions, the estimated values of p_1, p_2, p_3, and p_4 were then used in the model equation while segment traffic density $\rho_{k,j}$ and throughputs were computed from the resulting traffic flows. The model was used to predict density and speed of the middle segment by using the available data from the adjoining segments ($x_{k(j-1)}$, ρ_{kj}, $\rho_{k(j+1)}$, and throughputs). This model was found to be particularly effective in predicting speed of traffic flow and density in one segment of the freeway over 15-min intervals. The time interval of 4.5 s was adequate for traffic responsive control. The single-segment density prediction results from the model and actual density are shown in Figure 8.8. The single-segment speed prediction results from the model and actual segment speed are shown in Figure 8.9. The results show that the final model with the parameter values estimated by the above procedures predicted traffic conditions (density and speed) satisfactorily.

[4]The nominal speed is about where USDOT (United States Department of Transportation) formulas predict maximum throughput, which is in the order of 2000 vehicles per lane per hour. However, early studies by US traffic engineers in China found values closer to 30,000 vehicles per lane per hour in some cities. The difference was that those "vehicles" were bicycles and their speeds were around 12 mph.

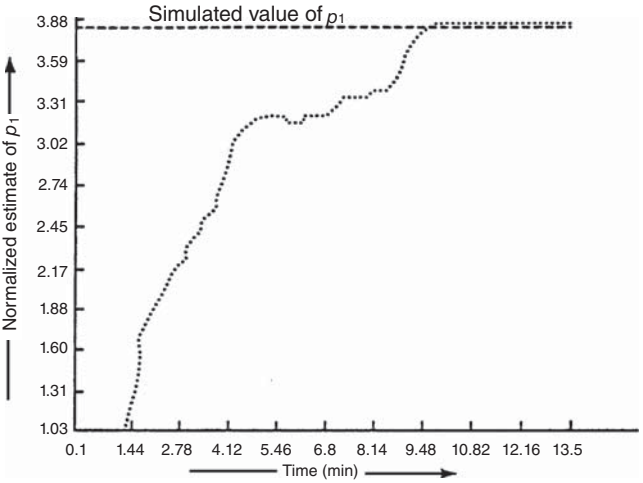


Figure 8.6 Estimation of reaction time.

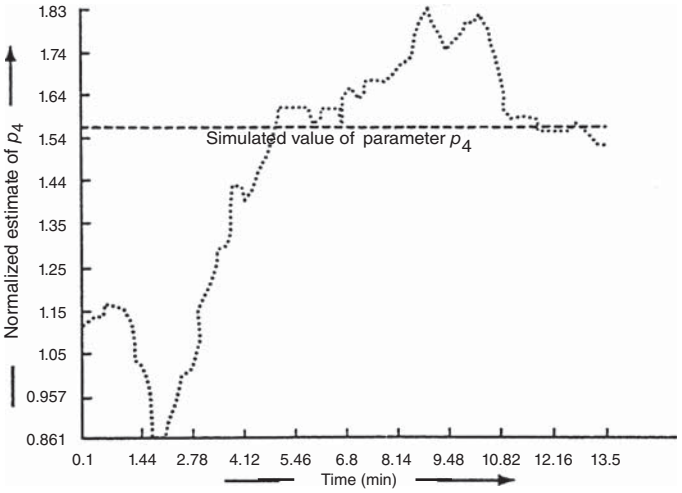


Figure 8.7 Estimation of sensitivity factor.

8.3.3.3 The Iterated Extended Kalman Filter (IEKF) Iteration is a long-established approach to solving many nonlinear problems, such as using Newton's method for solving nonlinear zero-crossing or minimization problems. There have been several iterated implementation techniques used in parts of nonlinear Kalman filtering, including temporal updates and observational updates in the iterated extended Kalman filter (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.

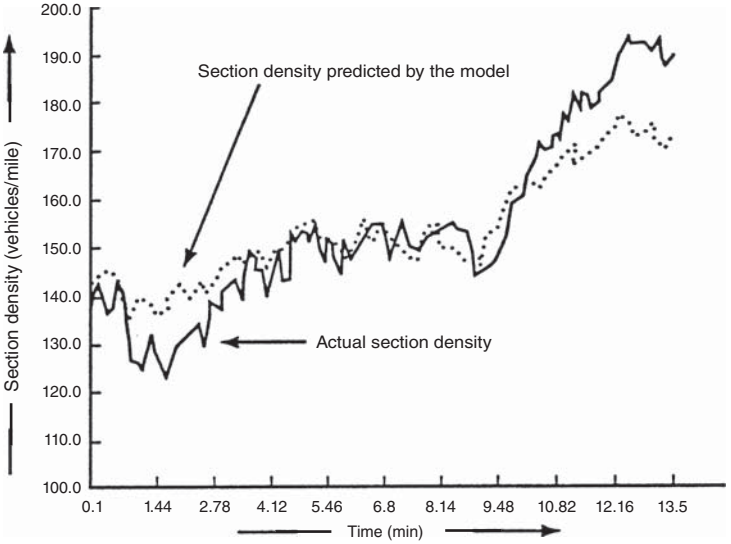


Figure 8.8 Single-segment density prediction.

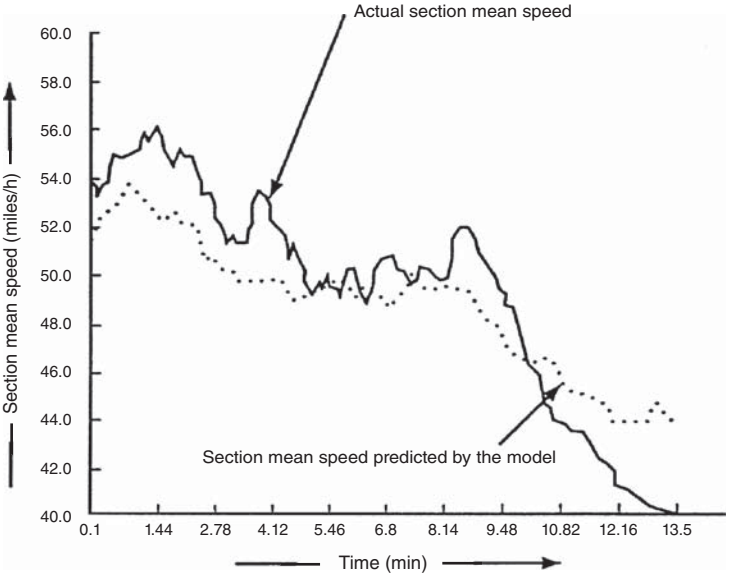


Figure 8.9 Single-segment speed prediction.

Iterated Measurement Update A performance comparison of nonlinear filtering methods by Lefebvre et al. [5] showed the IEKF to outperform the EKF, central difference filter (CDF), divided difference filter (DDF), and UKF for implementing the measurement update of the covariance matrix. Other studies have ranked sampling-based methods (e.g., UKF) higher overall but did not evaluate the observational update independently.

In the EKF, the measurement function $h(t)$ is linearized by partial differentiation,

$$H_k \approx \left.\frac{\partial h}{\partial x}\right|_{x=\hat{x}_k(-)} \tag{8.31}$$

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 EKF,

$$\hat{x}_k^{[0]} \stackrel{\text{def}}{=} \hat{x}_{k(+)}, \tag{8.32}$$

which is the zeroth iteration. The final a posteriori covariance of estimation uncertainty is only calculated after the last iteration, using the final iterated value of H.

Iteration proceeds for i=1, 2, 3, …

$$H_k^{[i]} = \left.\frac{\partial h}{\partial x}\right|_{x=x_k^{[i-1]}} \tag{8.33}$$

$$\overline{K}_k^{[i]} = P_{k(-)} H_k^{\mathrm{T}\,[i]} [H_k^{[i]} P_{k(-)} H_k^{[i]\mathrm{T}} + R_k]^{-1} \tag{8.34}$$

$$\tilde{z}_k^{[i]} = z_k - h_k(\hat{x}_k^{[i-1]}) \tag{8.35}$$

$$\hat{x}_k^{[i]} = \hat{x}_{k(-)} + \overline{K}_k^{[i]} \{\tilde{z}_k^{[i]} - H_k^{[i]} [\hat{x}_{k(-)} - \hat{x}_k^{[i-1]}]\}, \tag{8.36}$$

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 ϵ_{limit}. For example, any of the conditions

$$|\hat{x}_k^{[i]} - \hat{x}_k^{[i-1]}|_2 < \epsilon_{\text{limit}} \tag{8.37}$$

$$|\hat{x}_k^{[i]} - \hat{x}_k^{[i-1]}|_\infty < \epsilon_{\text{limit}} \tag{8.38}$$

$$(\hat{x}_k^{[i]} - \hat{x}_k^{[i-1]})^{\mathrm{T}} P_k^{-1}(-)(\hat{x}_k^{[i]} - \hat{x}_k^{[i-1]}) < \epsilon_{\text{limit}} \tag{8.39}$$

might be used as a condition to cease iterating, where the threshold ϵ_{limit} has been chosen based on the 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:

$$P_{k(+)} = P_{k(-)} - \overline{K}_k^{[i]} H_k^{[i]} P_{k(-)}, \tag{8.40}$$

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 recordkeeping of the Riccati equation.

Example 8.6 (Nonlinear Bearing-only Measurement Model) As an example of a nonlinear measurement function, consider the sensor geometry illustrated in Figure 8.10, where the angular measurement

$$z_k = \theta_k + v_k \tag{8.41}$$

$$= \arctan\left(\frac{x_k}{d}\right) + v_k \tag{8.42}$$

$$= h(x_k) + v_k \tag{8.43}$$

is the angle θ (plus zero-mean white noise $\{v_k\}$), a nonlinear function of the state variable x_k.

If the partial derivative is used as an approximation for the measurement sensitivity matrix,

$$H \approx \left.\frac{\partial h}{\partial x}\right|_x \tag{8.44}$$

$$= \frac{d}{d^2 + x^2}, \tag{8.45}$$

a function of the value of x at which it is evaluated.

Figure 8.11 is a plot of 100 Monte Carlo implementations of the IEKF with five iterations on this problem with the offset distance $d = 1$. The problem parameters used are

$d = 1$

$R = 10^{-6}$ (1 mrad RMS sensor noise)

$P = 1$ (mean-squared estimation uncertainty)

$x = 1/2$ (true value of state variable)

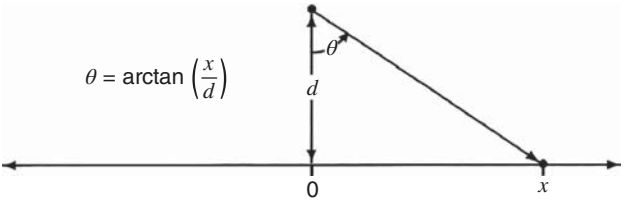


Figure 8.10 Nonlinear measurement model.

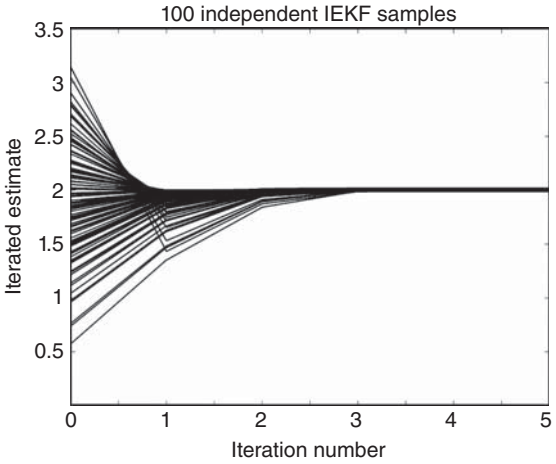


Figure 8.11 100 Monte Carlo simulations of IEKF convergence.

$$z = \arctan(x) + v, v \in \mathcal{N}(0, \sqrt{R}) \text{ (sample measurement)}$$
$$\hat{x} = x + w, w \in \mathcal{N}(0, \sqrt{P}) \text{(sample estimate)}$$

The plot in Figure 8.11 shows how the 100 randomly sampled initial estimates converge in the IEKF implementation of Equations 8.33 through 8.36 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 estimation errors and random measurement noise. The results shown in Figure 8.11 all converge, which is not that 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.

8.3.4 Bounding RMS Linearization Errors

The relative magnitudes of filtering errors due to linear approximation can be analyzed in a number of ways, including

1. Monte Carlo analysis, by which the means and covariances obtained by the EKF can be compared to the results of Monte Carlo simulation. This approach was used as part of the vetting process for Kalman filtering before it was accepted for use on the Apollo navigation problem. It is necessary to assume that all the probability distributions involved are completely known, however.

2. Propagation of multiples of the "one-sigma" points of the equiprobability ellipse through the nonlinearity and comparing the results to the linearized approximation.

In the second of these, linearization errors in the computation of means and covariances can be assessed by comparing the $N\sigma$ points of a the linearized approximation with values obtained by nonlinear simulation for $N \geq 3$. The "sigma points" in this case are the products of the eigenvectors of the covariance matrix of state uncertainty, multiplied by the square roots of the associated eigenvalues. These can be computed using singular value decomposition (SVD, `svd` in MATLAB) of the initial covariance matrix to obtain its eigenvalue−eigenvector decomposition as

$$P = \sum_i \sigma_i^2 e_i e_i^{\mathrm{T}}, \tag{8.46}$$

where the σ_i^2 are the singular values of P, the e_i are the associated eigenvectors, and the vectors

$$\pm N\ \sigma_i e_i$$

are multiples of the "sigma points" of the distribution.

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 dominated by the 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, which can be estimated using the covariances from linearized Kalman filtering (Section 8.3.3.1). The ranges can be specified in units 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 $N \approx 3$ or a bit more, 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

$$(\delta x)^{\mathrm{T}} P^{-1} (\delta x) \leq nN^2, \tag{8.47}$$

$$\varepsilon = \underbrace{\phi(\hat{x} + \delta x) - \left[\phi(\hat{x}) + \left.\frac{\partial \phi}{\partial x}\right|_{\hat{x}} \delta x\right]}_{\text{Approximation error}}, \tag{8.48}$$

$$\varepsilon^{\mathrm{T}} Q^{-1} \varepsilon \ll n, \tag{8.49}$$

where n is the dimension of the state vector. 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

$$(\delta x)^{\mathrm{T}} P^{-1} (\delta x) \leq nN^2, \tag{8.50}$$

$$\varepsilon_h = \underbrace{h(\hat{x} + \delta x) - \left[h(\hat{x}) + \left. \frac{\partial h}{\partial x} \right|_{\hat{x}} \delta x \right]}_{\text{Approximation error}}, \tag{8.51}$$

$$\varepsilon_h^{\mathrm{T}} R^{-1} \varepsilon_h \ll \ell, \tag{8.52}$$

where ℓ s the dimension of the measurement vector. The value of estimation uncertainty covariance P used in Equation 8.50 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.

8.3.4.1 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 8.12 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 SVD[5] of a covariance matrix P:

$$P = U \Lambda U^{\mathrm{T}} \tag{8.53}$$

$$= \sum_{i=1}^{n} u_i \sigma_i^2 u_i^{\mathrm{T}} \tag{8.54}$$

$$\delta x_i = N \sigma_i u_i, \; 1 \leq i \leq n \tag{8.55}$$

$$\delta x_{2n+i} = -N \sigma_i u_i, \; 1 \leq i \leq n, \tag{8.56}$$

[5]Implemented by the MATLAB function `svd`.

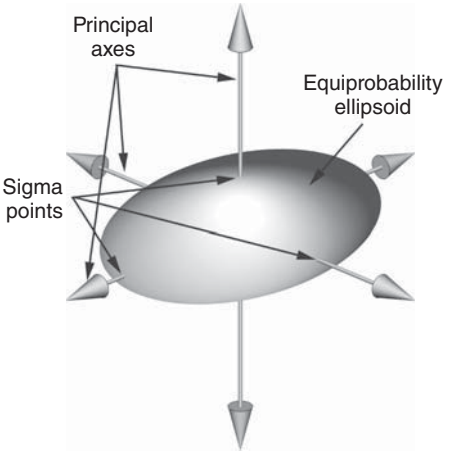


Figure 8.12 Principal axes of equiprobability ellipsoid.

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 8.55 and 8.56 will yield $2n$ perturbation samples, where n is the dimension of x.

Conditions 8.47 and 8.50 depend on estimation uncertainty. The bigger the uncertainties in $\hat{x}$ are, the larger the perturbations must be for satisfying the quasilinearity constraints.

This approach is illustrated in the following example.

Example 8.7 (Satellite Pseudorange Measurements) 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 8.13.

The GNSS navigation solution for the location of the receiver antenna requires several such pseudoranges to satellites in different directions. Each satellite broadcasts its precise transmission time and 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.

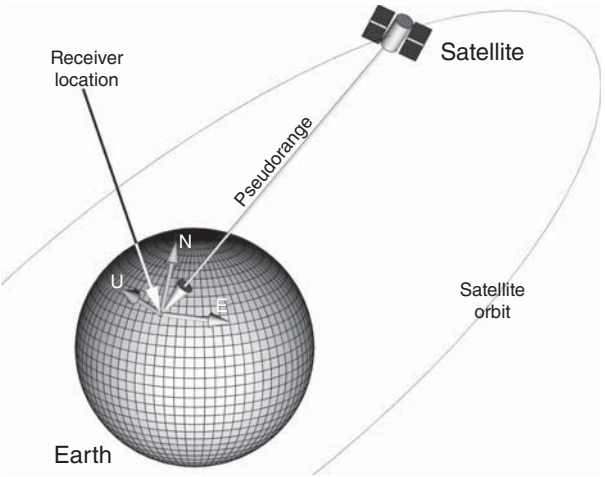


Figure 8.13 GNSS satellite pseudorange ρ.

One can use Equation 8.51 to determine whether the pseudorange measurement is sufficiently linear, 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 root-mean-square (RMS) position uncertainty is uniform in all directions, and

$$P = \begin{bmatrix} \sigma_{\text{pos}}^2 & 0 & 0 \\ 0 & \sigma_{\text{pos}}^2 & 0 \\ 0 & 0 & \sigma_{\text{pos}}^2 \end{bmatrix} \text{ (receiver position covariance)},$$

$$R = \sigma_\rho^2 \text{ (pseudorange noise variance)},$$

$$\sigma_\rho = 10 \text{ m (RMS pseudorange noise)},$$

$$R_{\text{sat}} = 2.66 \times 10^7 \text{m, satellite orbit radius},$$

$$R_{\text{rec}} = 6.378 \times 10^6 \text{m, receiver radius from earth center},$$

$$\lambda = 0^\circ,\ 30^\circ,\ 60^\circ, \text{ and } 90^\circ \text{ elevation of satellite above the horizon},$$

$$\alpha = \text{azimuth direction to the satellite},$$

$$\rho_0 = \sqrt{R_{\text{sat}}^2 - R_{\text{rec}}^2 \cos(\lambda)^2} - R_{\text{rec}} \sin(\lambda) \text{ (nominal pseudorange)},$$

$$X_{\text{sat}} = \rho_0 \begin{bmatrix} -\cos(\lambda)\cos(\alpha) \\ -\cos(\lambda)\sin(\alpha) \\ \sin(\lambda) \end{bmatrix} \text{(satellite position in east-north-up coordinates)},$$

$$\hat{x} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \text{(estimated receiver position in east-north-up coordinates)},$$

$$X_{\text{rec}} = \delta x$$

$$= \begin{bmatrix} \delta x_E \\ \delta x_N \\ \delta x_U \end{bmatrix} \text{(true receiver position in east-north-up coordinates)},$$

$$\rho = |X_{\text{rec}} - X_{\text{sat}}| \text{(true pseudorange)},$$

$$= h(\hat{x} + \delta x),$$

where the last equation is the nonlinear formula for pseudorange as a function of perturbations δx of receiver antenna position in east-north-up coordinates.

The linear approximation for the sensitivity of pseudorange to antenna position is

$$H \approx \left.\frac{\partial h}{\partial \delta x}\right|_{\delta x = 0}$$

$$= \begin{bmatrix} \cos(\lambda)\cos(\alpha) & \cos(\lambda)\sin(\alpha) & -\sin(\lambda) \end{bmatrix}.$$

In this case, the nonlinear approximation error defined by Equation 8.51 will be a function of the satellite elevation angle α and the perturbation δx.

The RMS nonlinearity error for the six perturbations

$$\delta x = \begin{bmatrix} \pm 3\sigma_{\text{pos}} \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ \pm 3\sigma_{\text{pos}} \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ \pm 3\sigma_{\text{pos}} \end{bmatrix}$$

is plotted in Figure 8.14 as a function of σ_{pos} for elevation angles $\alpha = 0^\circ,\ 30^\circ,\ 60^\circ$ 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 dashed horizontal line represents the RMS pseudorange noise, indicating that nonlinear approximation errors are dominated by pseudorange noise for RMS position uncertainties less than ≈ 7 km.

Typical RMS position errors in GNSS navigation are in the order of 1–100 m, which is well within the quasilinear range. This indicates that extended Kalman filtering with analytical partial derivative approximations is certainly justified for GNSS navigation applications. Furthermore, because it uses simple analytical partial derivative formulas, it will more efficient than the UKF.

8.3.5 Multilocal Linearization for Detection

This is a general method for overcoming initial nonlinearities in applications for which initial estimation uncertainties are significantly greater than those after a few observations have been processed and for which linear approximation errors tend

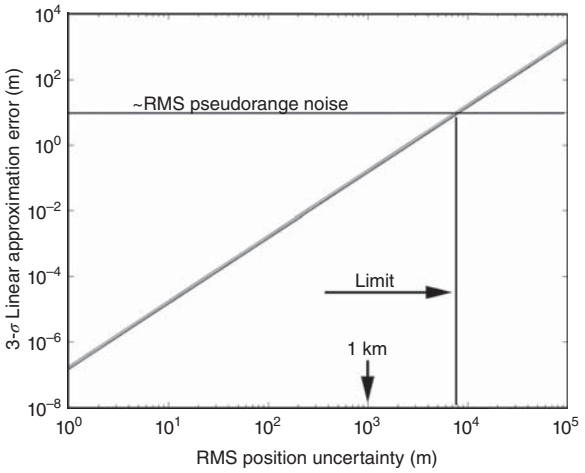


Figure 8.14 Linearization error analysis of pseudorange measurements.

to dominate [6]. It provides relief for that brief period when linearization errors are a significant problem, eliminating the extra computational burden when they are not.

8.3.5.1 Detection versus Tracking The Kalman filter is essentially a *tracking algorithm*. Given an initial estimate of the state of a system and the accompanying covariance of estimation uncertainty, it uses the sequence of measurements to track the state of the system thereafter. Nonlinear approximations to Kalman filtering generally depend on the initial uncertainties being sufficiently small that nonlinear approximation errors are not likely to be statistically significant.

Detection is the problem of obtaining that initial estimate to start tracking. It can lead to implementation problems from two sources:

1. In applications where there is inadequate information to start with, the effective covariance matrix would be infinite. If arbitrarily large values are used, instead, this can easily lead to numerical stability problems in the Kalman filter. This is a fundamental problem in Kalman filtering which is best addressed by using an alternative detection algorithm to initialize the estimate and its covariance or by switching to information-based estimation (which can cope with zero initial information) until sufficient estimation accuracy is attained.
2. If nonlinear approximation methods are being used, then nonlinear approximation errors are more likely to be unacceptable with large initial uncertainties. This problem has been addressed by using a multitude of potential initial guesses $\hat{x}_0$ and something called *Schweppe Gaussian likelihood ratio detection* [7, 8] for selecting the most likely of those initial guesses.

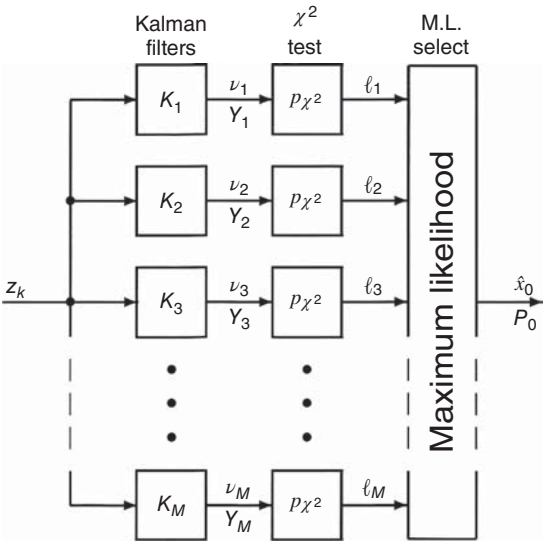


Figure 8.15 Multihypothesis detection/initialization.

Statistical Detection and Selection Schweppe likelihood ratio detection is derived in Section 3.6.5 as a method for deciding which of the two contending hypotheses is more likely, based on the analysis of the filter innovations.

For the nonlinear initialization problem, the approach can be generalized to decide which of a multitude of locally linearized initial guesses of the initial value of a Kalman filter state variable is the most likely, and that multitude of guesses is used for the purpose of reducing initial errors due to nonlinearities. The general approach is shown schematically in Figure 8.15, in this case using a multihypothesis chi-squared test to select the most likely initial guess.

If a Kalman filter is properly modeled, the innovations will be statistically independent, so that the probability of a sequence of innovations is just the product of the individual probabilities, and the logarithm of this product will be the sum of the logarithms of the individual probabilities.

In practice, this is often implemented by summing the logarithms of the *likelihoods* and selecting the largest, or comparing the relative closeness of the mean of the Kalman filter innovations to its expected value ℓ (the dimension of the innovations vector), or the closeness of the normalized mean $\overline{\nu}/\ell$ to 1, or by employing any combination of these measures.

8.3.5.2 Partitioning the Detection Domain The "detection domain" is the region in which the initial value of the state vector may lie. If it can be partitioned into a reasonable number of smaller domains to suppress linearization errors, then an independent Kalman filter can be initialized with a value in each of those subdomains. This depends on the detection domain being finite (or "compact" in topological terms), but there are problem attributes which lend themselves to this.

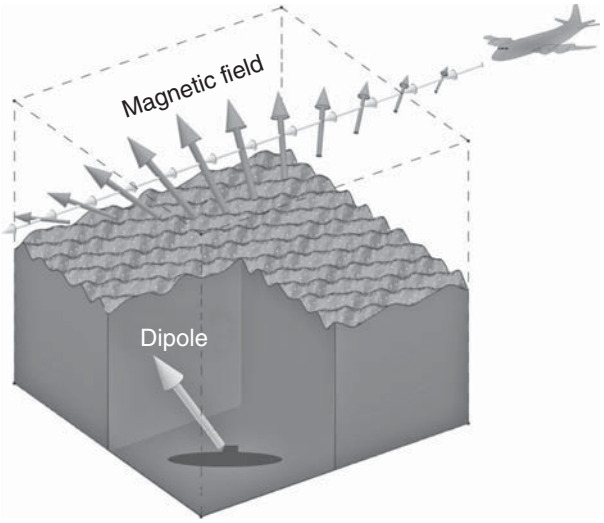


Figure 8.16 Airborne magnetic detection of dipoles.

Sensor Range Limitations In applications using sensors with limited range, the detection region could be within the range domain at which any signal might be discernible above the "noise floor." Radar, lidar, capacitance sensors, and magnetic sensors, for example, generally have output signals that tend to fall off as some inverse power of range.

The magnetic field sensor modeled in Reference 7, used for detecting and locating magnetic dipoles (Figure 8.16), is decidedly nonlinear and also has a limited detection range. In that case, a small number of subdomains can be chosen such that the nonlinearity errors are manageable within each subdomain, and a separate Kalman filter can be used within each subdomain.

The problem then becomes one of monitoring the performance of each Kalman filter and selecting one that estimates a reasonable magnetic moment with acceptable uncertainties.[6]

Subdividing Compact Manifolds In other applications, the full domain may be bounded naturally. Attitude, for example, can be represented by three angular variables and is commonly represented by unit quaternions, which parametrize the surface of the 3-sphere in 4-space in much the same way as unit vectors in 3-space parametrize the 2-sphere. Figure 8.17 represents a covering of the 2-sphere with 1-spheres (circles), in much the same way as the surface of the 3-sphere of the attitude domain might be covered by 2-spheres (which we are unable to depict in a two-dimensional drawing).

Attitude estimation is a notoriously nonlinear estimation problem [9], and being able to partition the search space into smaller domains can keep the influence of

[6]In practice, of course, the implementation does tend to be a bit more complicated.

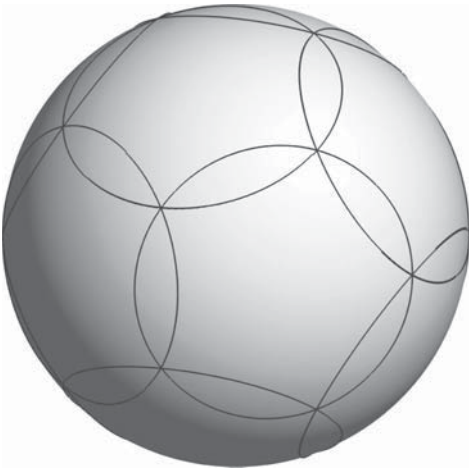


Figure 8.17 Covering the surface of the 2-sphere with 1-spheres.

nonlinear approximation errors down to levels that can be tolerated with an EKF or UKF, for example.

8.4 SAMPLE-AND-PROPAGATE METHODS

These methods for nonlinear propagation of probability densities have been applied to nonlinear extensions of the Kalman filter for propagating means and covariance matrices, including effects of nonlinear dynamics and nonlinear measurements. These are essentially perturbation methods, because they use samples as initial values for trajectories that are perturbations from the trajectory of the mean.

The EKF may also use perturbations for nonlinear propagation of means and expected measurements, but it only uses the perturbed trajectories to compute numerical partial derivatives. The sample-and-propagate methods use perturbed trajectories to approximate nonlinear transformations of the means and covariances.

Sampling essentially transforms a probability density distribution on a continuous domain into one on a discrete domain, by using the probability densities at a discrete set of points to represent the continuous distribution. In so doing, what may not have been computable with the continuous function becomes computable with the sampled set of numbers.

8.4.1 Assessing Performance

The problem of nonlinear propagation of just the means and covariances of a distribution—by any method—is ill-posed, in that there is no unique solution. The problem is that nonlinearities couple higher order moments into the means and covariances, and these higher order moments are not tracked. There is an unlimited

number of distributions with the same mean and covariance, all with different higher order moments. Knowing the nonlinear transformation is of no use if one does not know the higher order moments.

Properly designed sample-and-propagate methods for estimating the mean and covariance will always produce the exact solution when the transformation of the variate is linear. Beyond that, there is no exact solution—and just assessing performance can be complicated. It requires specifying all the probability distributions involved, including that of the initial state vector and all the dynamic and measurement error sources involved, and using that information in propagating the full state vector probability through the filtering history. If all the probability distributions involved can be specified and simulated, this might be achieved through Monte Carlo analysis of many filter history instances for a specific nonlinear model. The result would be an assessment of performance on a specific nonlinear model, with specific probability distributions—which can be very useful for "tuning" any parameters of the sample-and-propagate implementation to improve performance on a specified application. Still, one has to be specific about these attributes of the application in specifying the performance of a particular sample-and-propagate method.

8.4.2 Monte Carlo Analysis

This started as a general-purpose method for estimating the evolution over time of a probability distribution. It involves selecting a sufficient number of representative random samples and using simulation to propagate these samples according to the dynamic model for the system. It represents probability density as sample density. The resulting distribution of propagated samples can then provide an estimate of the propagated probability distribution.

It has also been used for assessing probability distributions of estimation error by running many instances of the estimator history with sample input histories.

8.4.2.1 Origins The essential idea behind Monte Carlo analysis is the way experienced gamblers learn their craft: by calculating probabilities based on observed outcomes. The formalization of this as a mathematical process can be traced to the seventeenth century Bernoulli brothers, but its very name and its computer implementation can be traced to experiences of mathematician Stanislaw Ulam (1909–1984) when he was recovering from a bout with encephalitis in 1946 [10]. He was playing a card game called *Canfield solitaire* to while away the hours of recovery and began to wonder about the probability of being dealt a winning hand. It soon became apparent that the problem was not solvable with pencil and paper,[7] but might be approximated by random sampling and simulation. When he returned to work at the Los Alamos National Laboratory, he had both the means for such an approach

[7]The number of distinct shufflings of a 52-card deck is $52! \approx 8 \times 10^{67}$. The odds of winning are not good. The game had been introduced by Richard A. Canfield (1855–1914), an American gambling parlor owner who was able to profit from offering a $500 prize for winning, after paying $50 up-front for a deck to try it on.

(the ENIAC computer) and the motivation (solving the neutron diffusion problem for nuclear devices). Ulam worked with John von Neumann (1903–1957) and Nicholas Metropolis[8] (1915–1999), among others, [38] in reducing it to practice.

As proposed by Ulam, the idea is to randomly select representative points from an initial probability distribution and let them do their own thing, then calculate the resulting distribution statistics of interest. Although the *law of large numbers* may guarantee eventual conversion of such methods, convergence is not necessarily fast enough for all practical purposes. Over the years there have been several modifications of the procedures to speed up the process, and there is a vast literature on the design and use of such methods in statistical testing.

8.4.2.2 Random Sampling Random sampling depends on knowing the probability distribution involved and having a method for generating random samples from that distribution. The most common methods are for Gaussian distributions or uniform distributions, but random number generators for uniform scalar distributions can be used to generate random samples for any distribution for which the cumulative probability distribution $P(\cdot)$ and its inverse are known. Given a value $x_{[i]}$ from a uniform distribution between 0 (zero) and 1, the corresponding sample from the distribution represented by $P(\cdot)$ is $y_{[i]} = P^{-1}(x_{[i]})$.

8.4.2.3 Democratic Sampling This uses the inverse cumulative probability function to generate a set of N representative samples, each representing (in some way) one Nth of the population.

Example 8.8 (Arctangent Transformation of Gaussian Variate) The arctangent is an analytic function with power series expansion

$$\arctan(x) = \sum_{k=0}^{\infty} \frac{(-1)^k}{2k+1} x^{2k+1},$$

convergent everywhere on the real line. It is called an *odd function* in that its power series expansion has only odd nonzero coefficients. It maps the infinite domain of the zero-mean univariate Gaussian distribution ($-\infty < x < +\infty$) into a finite range ($-\pi/2 \leq \arctan(x) \leq +\pi/2$).

The objective is to test how different nonlinear approximations perform in updating the means and covariances after the variate is transformed as

$$y(x) = \arctan(a\, x),$$

where a is an arbitrary positive scaling parameter.

[8]Metropolis' contributions to the method may have included naming the approach after the gambling casino at Monaco, where Ulam's uncle allegedly "played" [11].

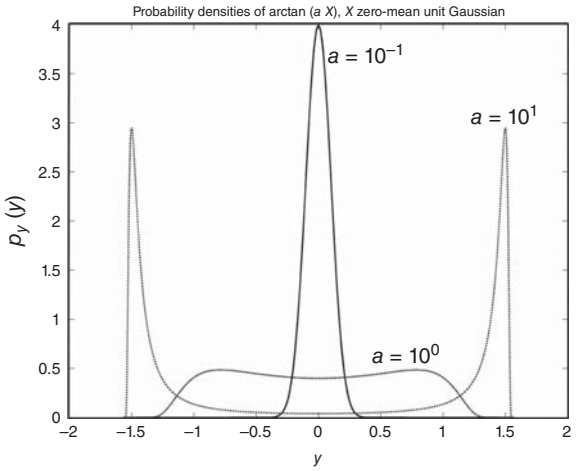


Figure 8.18 Probability densities of $y = \arctan(ax), x \in \mathcal{N}(0, 1)$.

If X is zero-mean unit-normal (Gaussian), the transformed variate $Y \stackrel{\text{def}}{=} \arctan(a\,X)$ will have probability density function

$$p_y(y) = \frac{1}{a\sqrt{2\pi}}\{1 + [\tan(y)]^2\} \exp\left(-\frac{1}{2}\left[\frac{\tan(y)}{a}\right]^2\right). \tag{8.57}$$

This probability density function is nonzero for

$$-\frac{\pi}{2} < y < \frac{\pi}{2}$$

and zero elsewhere. It has the shapes shown in Figure 8.18 for various values of the positive parameter a. These resemble a Gaussian distribution when $a \ll 1$, but are decidedly non-Gaussian for $a \gg 1$.

8.4.2.4 Applications to Kalman Filtering When these methods are used in Kalman filtering for nonlinear propagation of means and covariances of distributions, the statistical parameters required for computing the Kalman gain, the fundamental problem is that one does not know the initial probability distribution—except for its mean and covariance.

This has led to the development of some specialized sample-and-propagate methods for approximation under this constraint.

8.4.3 Particle Filters

Particle filters propagate representative samples of the estimated distribution, which can be viewed as "particles" entrained in the dynamic flow of the distribution [39]. It is generally an improvement over random sampling, because it samples

according to what statistics are to be estimated and the character of the nonlinearities involved. In the case of propagating the covariance matrix of a distribution, the selection of samples is biased to make the estimate converge faster than by using random sampling. It has also been called a *sequential Monte Carlo* method [37], because the sampling may change as the solution progresses. "Importance sampling" uses criteria that indicate the relative importance of sampled values for a specific application. Some particle filtering also makes use of particle weights that can be "tuned" to improve performance for a specific application—something that has been incorporated into the UKF, as well.

8.4.3.1 General Nonlinear Model An abstract general-purpose nonlinear model for the estimation problem can be represented in the form

$$x_k = f_{k-1}(x_{k-1}) + w_{k-1} \text{ (nonlinear state dynamics)} \tag{8.58}$$

$$z_k = h_k(x_k) + v_k \text{(nonlinear measurement)}, \tag{8.59}$$

where the functions $f_{k-1}(\cdot)$ and $h_k(\cdot)$ are known. Numerical evaluation of the function $f_{k-1}(x_{k-1})$ may be a simple function call, or it may require integrating a trajectory with x_{k-1} as its initial value.

8.4.3.2 Particle Sampling In practice, the ith sample (or "particle") $\mathcal{S}_{(k-1)\,[i]}$ (sample for x_{k-1}) is a known perturbation $\delta_{\,[i]}$ of the mean $\hat{x}_{k-1}$:

$$\mathcal{S}_{(k-1)\,[i]} = \hat{x}_{(k-1)(+)} + \delta_{\,[i]}, \tag{8.60}$$

where the perturbations are chosen to best represent the nonlinear transformation of the mean $\hat{x}$ and covariance P from the $(k-1)$th discrete-time epoch to the kth epoch. The predicted mean $\hat{x}_{k(-)}$ and covariance $P_{k(-)}$ are then estimated from the distribution of the transformed samples

$$\mathcal{S}_{k\,[i]} = f_{k-1}(\mathcal{S}_{(k-1)\,[i]}). \tag{8.61}$$

8.4.3.3 Solution These transformed samples can then be transformed through $h_k(\cdot)$ to obtain the nonlinear approximation to the covariance of sensor output variation due to state uncertainty (HPH^{T}) and the *cross-covariance* between sensor output variation and state vector estimation error (PH^{T}) used in computing the Kalman gain.

8.4.4 Sigma-Point Filters

8.4.4.1 Sampling These methods generally focus on perturbations located at critical points on the surface of the one-sigma equiprobability ellipse of an equivalent Gaussian distribution with the same mean and covariance as the estimate.

These use specific statistical parameters of the a posteriori estimate $\hat{x}_{(k-1)(+)}$ and its covariance matrix $P_{(k-1)(+)}$ to select the samples at time t_{k-1}. In the most straightforward case, sample values are determined from the estimated mean $\hat{x}$ and the eigenvalue–eigenvector decomposition of the covariance matrix P as

$$P = \sum_i \lambda_i e_i e_i^{\mathrm{T}}, \tag{8.62}$$

where the λ_i are positive numbers and the e_i are mutually orthogonal unit vectors such that

$$Pe_i = \lambda_i e_i. \tag{8.63}$$

The "positive" sigma-point sample values are then

$$S_i = \hat{x} + \sqrt{\lambda_i} e_i. \tag{8.64}$$

There are n of these for an $n \times n$ covariance, but the sampling may use both positive and negative perturbations, numbering $2n$ in all. Counting the "zero perturbation" (i.e., $\hat{x}$), the approach requires either $n+1$ or $2n+1$ applications of the dynamic model function $f_{k-1}(S_i)$.

For Gaussian distributions, the e_i correspond to the principal axes of the equiprobability ellipse. They can be calculated by using the *singular value decomposition* (calculated by the MATLAB function `svd`):

$$P = UD_\lambda U^{\mathrm{T}} \tag{8.65}$$

$$U = [e_1|e_2|e_2|\cdots|e_n] \tag{8.66}$$

$$D_\lambda = \mathrm{diag}(\lambda_1,\ \lambda_2,\ \lambda_3,\ \ldots,\ \lambda_n) \tag{8.67}$$

$$\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \cdots \geq \lambda_n \geq 0,$$

where U is an orthogonal matrix. The only thing unique about this method of decomposition is that the nonnegative singular values are generated in descending order.

The offsets of the sigma points from the mean in this case can be calculated as

$$\delta_{[i]} = \sqrt{\lambda_i} e_i, \tag{8.68}$$

in which case the matrix

$$\Gamma = [\delta_{[1]}|\delta_{[2]}|\delta_{[3]}|\cdots|\delta_{[n]}] \tag{8.69}$$

is a modified Cholesky factor of P. That is,

$$\Gamma\Gamma^{\mathrm{T}} = P. \tag{8.70}$$

The perturbed nonlinear trajectories in this case would be those with initial values

$$S_{(k-1)\,[i]} = \hat{x}_{(k-1)(+)} + \delta_{\,[i]}. \tag{8.71}$$

Figure 8.12 shows an example of such sigma points on an equiprobability surface of a Gaussian distribution in three dimensions. In nonlinear transformations of the state vector, however, error distributions do not necessarily remain Gaussian. Sigma-point filtering may also be defined in terms of sample points that are not necessarily related to the eigenvectors of the covariance matrix. They may, for example, be defined by the columns of any generalized Cholesky factor Γ of P, which would be any solution Γ of $\Gamma\Gamma^{\mathrm{T}} = P$.

8.4.4.2 Propagation Sigma-point filtering has the potential for using different propagation methods for different classes of sigma points, depending on the respective magnitudes of the associated standard deviations. This allows for additional samples to be used for propagating the larger perturbations. However, the propagation error will generally depend on the magnitude of the higher order variations, as well as the magnitude of perturbation.

8.4.4.3 Symmetric Square Roots of Covariance Matrices The diagonal matrix D_λ of Equation 8.67 has nonnegative values and a matrix "square root"

$$\sqrt{D_\lambda} = \mathrm{diag}(\sqrt{\lambda_1},\ \sqrt{\lambda_2},\ \sqrt{\lambda_3},\ \ldots,\ \sqrt{\lambda_n}) \tag{8.72}$$

such that for the symmetric matrix

$$S \overset{\mathrm{def}}{=} U\sqrt{D_\lambda}U^{\mathrm{T}}, \tag{8.73}$$

$$S^2 = S \times S \tag{8.74}$$

$$= P. \tag{8.75}$$

That is, S is a true matrix square root of P. Because it is symmetric, it is also a *generalized Cholesky factor*, and a symmetric one, to boot. However, it turns out that sigma-point-like filters are not overly sensitive to the form or the "square-root" matrix used.

8.5 UNSCENTED KALMAN FILTERS (UKF)

We say "filters" because there is a family of filters using different data structures, depending on the relative need for speed and the degree of nonlinearity, and different sample weightings that can be "tuned" for improved performance on a given application.

We say "unscented" because that is what Jeffrey Uhlmann named them. (See

```
http://www.ieeeghn.org/wiki6/index.php/First-Hand:The_Unscented_
  Transform
```

for an interview with Jeffrey Uhlmann on the subject.)

Uhlmann started development of the UKF in the 1990s by analyzing the performance of different sigma-point sampling and weighting strategies on a nonlinear estimation problem in robotics. His analysis showed no significant variation in performance related to the particular "square-root" factorization of the initial covariance matrix and established some weighting strategies that can be "tuned" to improve performance for different applications. The results of this and collaborative studies was a suite of nonlinear Kalman filter extensions introduced by Simon J. Julier, Jeffrey K. Uhlmann, Hugh F. Durrant-Whyte, and others [12–24], all based on a core methodology for approximating nonlinear transformations of the mean and covariance.

These filter implementations use sample-and-propagate methods for nonlinear propagation of state variables and nonlinear measurements, much as the EKF uses perturbations for numerical approximations of partial derivatives. The UKF makes more intelligent use of the perturbations, so it has about the same computational requirements as the EKF, but improved performance in many applications. These attributes favor the UKF for consideration in many nonlinear filtering applications.

The various implementations of the UKF are a subclass of sigma-point filters and equivalent to linear regression Kalman filters [25] (LRKF) in some cases [26].

8.5.1 The Unscented Transform (UT)

This is the core method for propagating means and covariances, including the cross-covariance between the state and measurement. It replaces the Riccati equation in Kalman filtering and changes the way the Kalman gain is computed.

8.5.1.1 Covariance Factoring The unscented transform (UT) implements mean and covariance propagation using the column vectors of a Cholesky decomposition of the covariance matrix. Like many of the other sample-and-propagate filters, it uses weightings of the columns of the Cholesky factors to allow for some fine-tuning to improve performance on specific applications. These weightings are constrained to be exact when the problem is linear but may otherwise be varied to adapt to problem-specific nonlinearities.

The Cholesky decomposition algorithm is implemented in the built-in MATLAB function `chol`, which yields a lower triangular solution $C_{(k-1)(+)}$ of

$$P_{(k-1)(+)} = C_{(k-1)(+)} C^{\mathrm{T}}_{(k-1)(+)} \tag{8.76}$$

with nonnegative diagonal elements.

8.5.1.2 Sampling Strategies Major sampling and weighting strategies used in the UT are summarized in Table 8.1.

Numbers of Samples The greatest number of sampling perturbations used in the UT are the zero perturbation (i.e., the mean itself) and the signed columns $\pm c_{[i]}$ of $C_{(k-1)(+)}$. That is, the initial conditions of the simulated trajectories will be

$$\mathcal{S}_{(k-1)(+)[i]} = \begin{cases} \hat{x}_{(k-1)(+)}, & i = 0 \\ \hat{x}_{(k-1)(+)} + c_{[i]}, & 1 \le i \le n \\ \hat{x}_{(k-1)(+)} - c_{[i-n]}, & n+1 \le i \le 2n \end{cases} \tag{8.77}$$

and the terminal values will be

$$\mathcal{S}_{k(-)[i]} = f_{k-1}(\mathcal{S}_{(k-1)(+)[i]}), 0 \le i \le 2n. \tag{8.78}$$

However, the subset $\mathcal{S}_{(k-1)(+)[i]}$ for $0 \le i \le n$ may also be used.

TABLE 8.1 Unscented Transform Sample Weights

Sampling Strategy	Sample Size*	Sample Values†	Sample Weights‡	Index Values
Simplex	$n+2$	$\mathcal{S}_0 = \hat{x}$	$0 \le \mathcal{W}_0 \le 1$ (can be varied)	
		$\mathcal{S}_{[i]} = \mathcal{S}_0 + \gamma_{[i]}$	$\mathcal{W}_1 = 2^{-n}(1 - \mathcal{W}_0)$	
		$\Gamma = C_{xx}\Psi$	$\mathcal{W}_2 = \mathcal{W}_1$	
		$\Psi = [\psi_1\ \psi_2\ \cdots\ \psi_{n+1}]$	$\mathcal{W}_i = 2^{i-1}\mathcal{W}_1$	$3 \le i \le n+1$
		Algorithms for $\psi_{[i]}$ & $\mathcal{W}_i$ are in MATLAB function `UTsimplex`.		
Symmetric	$2n$	$\mathcal{S}_{[i]} = \hat{x} + \sqrt{n}\, c_{[i]}$	$\mathcal{W}_i = 1/(2n)$	$1 \le i \le n$
		$\mathcal{S}_{i+n} = \hat{x} - \sqrt{n}\, c_{[i]}$	$\mathcal{W}_{i+n} = 1/(2n)$	$1 \le i \le n$
	$2n+1$	$\mathcal{S}_0 = \hat{x}$	$\mathcal{W}_0 = \kappa/\sqrt{n+\kappa}$	
		$\mathcal{S}_{[i]} = \hat{x} + \sqrt{n+\kappa}\, c_{[i]}$	$\mathcal{W}_i = 1/[2(n+\kappa)]$	$1 \le i \le n$
		$\mathcal{S}_{i+n} = \hat{x} - \sqrt{n+\kappa}\, c_{[i]}$	$\mathcal{W}_{i+n} = 1/[2(n+\kappa)]$	$1 \le i \le n$
Scaled	$2n+1$	$\mathcal{S}_0 = \hat{x}$	$\mathcal{W}_0^{[\hat{y}]} = \lambda/(n+\lambda)$	
		$\lambda = \alpha^2(n+\kappa) - n$	$\mathcal{W}_0^{[P_{yy}]} = \mathcal{W}_0^{[\hat{y}]} + (1 - \alpha^2 + \beta)$	
		$\mathcal{S}_{[i]} = \hat{x} + \sqrt{n+\lambda}\, c_{[i]}$	$\mathcal{W}_i = 1/[2(n+\kappa)]$	$1 \le i \le n$
		$\mathcal{S}_{n+i} = \hat{x} - \sqrt{n+\lambda}\, c_{[i]}$	$\mathcal{W}_{n+i} = 1/[2(n+\kappa)]$	$1 \le i \le n$

$* n$ = dimension of state space.
† $c_{[i]}$ = ith column of a Cholesky factor C_{xx} of P_{xx}. $\gamma_{[i]}$ is the ith column of Γ.
‡ α, β, κ, and λ are "tuning" parameters.

Even smaller sample sizes were later obtained by using n-simplices.[9] However, the resulting matrix of simplex vectors is multiplied by a Cholesky factor of P to obtain the sampled values to be propagated for estimating the mean and covariance. This could be viewed as a form of "preweighting" of the Cholesky factor samples—before propagation.

8.5.1.3 Weighting Strategies Major strategies for computing weighted averages of propagated statistics are summarized in Table 8.1—and there are more.

The means, covariances, and cross-covariances of state and measurement uncertainty, given the perturbed-and-simulated values $\mathcal{S}_{k(-)i}$, will be weighted averages

$$\hat{x}_{k(-)} = \sum_i \mathcal{W}_{\hat{x},i}\mathcal{S}_{k(-)i} \tag{8.79}$$

$$= \Phi_{k-1}\hat{x}_{(k-1)(+)} \text{ when } f_{k-1}(x) = \Phi_{k-1}x \tag{8.80}$$

$$P_{k(-)} = \sum_i \mathcal{W}_{Pxx,i}(\mathcal{S}_{k(-)i} - \hat{x}_{k(-)})(\mathcal{S}_{k(-)i} - \hat{x}_{k(-)})^{\mathrm{T}} + Q_{k-1} \tag{8.81}$$

$$= \Phi_{k-1}P_{(k-1)(+)}\Phi_{k-1}^{\mathrm{T}} + Q_{k-1} \text{ when } f_{k-1}(x) = \Phi_{k-1}x \tag{8.82}$$

$$\hat{z}_k = \sum_i \mathcal{W}_{\hat{z},i}h_k(\mathcal{S}_{k(-)i}) \tag{8.83}$$

$$= H_k\hat{x}_{k(-)} \text{ when } f_{k-1}(x) = \Phi_{k-1}x \text{ and } h_k(x) - H_kx \tag{8.84}$$

$$P_{zz,k} = \sum_i \mathcal{W}_{Pzz,i}[h_k(\mathcal{S}_{k(-)i}) - \hat{z}_k][h_k(\mathcal{S}_{k(-)i}) - \hat{z}_k]^{\mathrm{T}} \tag{8.85}$$

$$= H_kP_{k(-)}H_k^{\mathrm{T}} \text{ when } h_k(x) = H_kx \tag{8.86}$$

$$P_{xz,k} = \sum_i \mathcal{W}_{Pxz,i}(\mathcal{S}_{k(-)i} - \hat{x}_{k(-)})[h_k(\mathcal{S}_{k(-)i}) - \hat{z}_k]^{\mathrm{T}} \tag{8.87}$$

$$= P_{k(-)}H_k^{\mathrm{T}} \text{ when } f_{k-1}(x) = \Phi_{k-1}x \text{ and } h_k(x) = H_kx, \tag{8.88}$$

where the weightings $\mathcal{W}_{\cdot,i}$ are constrained to obtain the same result as the Kalman filter when $f_{k-1}(\cdot)$ and $h_k(\cdot)$ are linear. Those constraints do not uniquely determine the weightings, however.

Also, if the variations in the functions f_{k-1} and h_k with respect to x at $\hat{x}$ are antisymmetric at $\hat{x}$, that is, for $i > n$

$$(\mathcal{S}_{k(-)i} - \hat{x}_{k(-)}) = -(\mathcal{S}_{k(-)(i-n)} - \hat{x}_{k(-)}) \tag{8.89}$$

$$[h_k(\mathcal{S}_{k(-)i}) - \hat{z}_k] = -[h_k(\mathcal{S}_{k(-)(i-n)}) - \hat{z}_k], \tag{8.90}$$

[9] An n-simplex sampling is a set of $n+1$ equidistant points in n-dimensional real space. They can be defined as the vertices of the set of points in $(n+1)$-dimensional space whose coordinates are nonnegative and add up to 1. For example, the 1-simplex is the two ends of a line segment, the 2-simplex is the three vertices of a triangle, and the 3-simplex is the four corners of a tetrahedron.

then the samples with $i > n$ are redundant. That is, only $n + 1$ samples of the state vector are required.

In fact, even if $h_k(\cdot)$ is not antisymmetric but $f_{k-1}(\cdot)$ is, only $n + 1$ samples of the state vector are required. The other samples for $h_k(\cdot)$ can be generated by using the antisymmetry of $f_{k-1}(\cdot)$ in Equation 8.89.

8.5.1.4 Weighting Conformity with the Linear Case The weights of the UT must yield the linear result when core function is linear.

For example, for means, for any $0 < \alpha \leq 1$, the weighting strategy

$$\mathcal{W}_{\hat{x},i} = \begin{cases} \alpha, & i = 0 \\ \dfrac{1-\alpha}{2n}, & i > 0 \end{cases} \tag{8.91}$$

is equivalent to the Kalman filter values when the problem is linear. That is,

$$\hat{x}_{k(-)} = \sum_{i=1}^{2n} \underbrace{\mathcal{W}_{\hat{x},i} H_k \{\hat{x}_{(k-1)(+)} + c_i\}}_{\mathcal{W}_{\hat{x},i} s_{k(-)i}} \tag{8.92}$$

$$= \left\{ \alpha + 2n \times \frac{(1-\alpha)}{2n} \right\} H_k \hat{x}_{(k-1)(+)} + \frac{(1-\alpha)}{2n} H_k \sum_{i=1}^{n} (c_i - c_i) \tag{8.93}$$

$$= H_k \hat{x}_{(k-1)(+)}, \tag{8.94}$$

the linear Kalman filter formula. The same weighting formula works for $\hat{z}_{k(-)}$. For $\alpha = 1$, the results are the EKF values.

The same sort of test can also be applied to the covariance weights.

8.5.2 UKF Implementations

Example 8.9 (Simplex Unscented Transform with Alternative Choleksy Factors) This is a demonstration of the UT "simplex" sampling strategy of Table 8.1, using the nonlinear measurement problem with initial mean and covariance

$$\hat{x} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{8.95}$$

$$P_{xx} = \begin{bmatrix} 84 & -64 & -32 & 16 \\ -64 & 84 & 16 & -32 \\ -32 & 16 & 84 & -64 \\ 16 & -32 & -64 & 84 \end{bmatrix}, \tag{8.96}$$

respectively.

All the UTs 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}:

$$C_{\mathrm{UT}} = \begin{bmatrix} 3.6956 & -7.4939 & -3.3371 & 1.7457 \\ 0 & 8.3556 & -1.4118 & -3.4915 \\ 0 & 0 & 5.9362 & -6.9830 \\ 0 & 0 & 0 & 9.1652 \end{bmatrix} \text{(upper triangular)} \tag{8.97}$$

$$C_{\mathrm{LT}} = \begin{bmatrix} 9.1652 & 0 & 0 & 0 \\ -6.9830 & 5.9362 & 0 & 0 \\ -3.4915 & -1.4118 & 8.3556 & 0 \\ 1.7457 & -3.3371 & -7.4939 & 3.6956 \end{bmatrix} \text{(lower triangular)} \tag{8.98}$$

$$C_{\mathrm{EV}} = \begin{bmatrix} -7 & -5 & 3 & 1 \\ 7 & 5 & 3 & 1 \\ 7 & -5 & -3 & 1 \\ -7 & 5 & -3 & 1 \end{bmatrix} \text{(eigenvalue}-\text{eigenvector)}, \tag{8.99}$$

and for the second-order nonlinear measurement function

$$z = h(x) \tag{8.100}$$

$$= \begin{bmatrix} x_2 x_3 \\ x_3 x_1 \\ x_1 x_2 \end{bmatrix}. \tag{8.101}$$

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

$$\hat{z} = \mathrm{E}_x\langle h(x)\rangle \tag{8.102}$$

$$= \frac{1}{(2\pi)^2 \det P_{xx}^{-1}} \int dx_4 \int dx_3 \int dx_2 \int dx_1 \; h(x) \; \exp\left(-x P_{xx}^{-1} x^{\mathrm{T}}/2\right) \tag{8.103}$$

$$= \begin{bmatrix} P_{2,3}\left(1 - P_{2,3}^2/P_{2,2}/P_{3,3}\right)^{-3/2} \\ P_{3,1}(1 - P_{3,1}^2/P_{3,3}/P_{1,1})^{-3/2} \\ P_{1,2}(1 - P_{1,2}^2/P_{1,1}/P_{2,2})^{-3/2} \end{bmatrix}. \tag{8.104}$$

That is, the true mean for an initial Gaussian distribution with zero mean depends on the covariance. This dependence is not represented in extended Kalman filtering.

The results, shown in Table 8.2, were generated by the MATLAB m-file `UTsimplexDemo.m`, using the MATLAB function `UTsimplex`, both of which are on the Wiley web site.

The EKF approximation values are also shown in Table 8.2, along with the exact Gaussian solution of Equation 8.104. So that noise covariance does not mask nonlinear error approximations, we also assume that the sensor noise covariance $R = 0$.

TABLE 8.2 Results from Example 8.9: Simplex unscented transform with Alternative Choleksy Factors

Nonlinear Measurement Problem

$$\hat{x} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \; P_{xx} = \begin{bmatrix} 84 & -64 & -32 & 16 \\ -64 & 84 & 16 & -32 \\ -32 & 16 & 84 & -64 \\ 16 & -32 & -64 & 84 \end{bmatrix}, \; z = h(x) = \begin{bmatrix} x_2 x_3 \\ x_3 x_1 \\ x_1 x_2 \end{bmatrix}, \; R = 0$$

Solution by Simplex Unscented Transform with $W_0 = 1/2$

Cholesky Factor	Upper Triangular	Lower Triangular	Eigen vector
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 Extended Kalman Filter Approximation

$$\hat{z} \approx h(\hat{x}) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \; H \approx \left.\frac{\partial h}{\partial x}\right|_{x=\hat{x}} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \; P_{zz} \approx H P_{xx} H^{\mathrm{T}} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Results generated by the MATLAB m-file `UTsimplexDemo.m`, using the MATLAB function `UTsimplex`.

Example 8.9 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} are not what we are used to in extended Kalman filtering of truly quasilinear problems. The initial application of extended Kalman filtering on the Apollo moon missions was able to achieve trajectory estimation accuracies in the order of tens of kilometers at distances in 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 very much 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 it is 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.

Example 8.10 (Symmetric Unscented Transform with Alternative Choleksy Factors) Results from using the symmetric UT sampling strategy with the same nonlinear measurement problem as Example 8.9 are shown in Table 8.3. These were generated by the MATLAB m-file `UTscaledDemo.m`, using the MATLAB function `UTscaled`, both of which are on the Wiley web site, and should be compared to the results in Table 8.2 for the simplex UT.

These results are not that different from those using the simplex UT and exhibit similar sensitivities to the choice of Cholesky factor. They are significantly better than the EKF results, however.

This example shows the importance of vetting any nonlinear filtering method on the intended application before proceeding with it.

8.5.2.1 Data Flow versus EKF A data flow diagram of the UKF with $2n + 1$ samples is shown in Figure 8.19. Note the similarity to the comparable EKF data flow diagram in Figure 8.2. With comparable computational effort, the UKF is able to use a more robust strategy for propagating covariances.

Example 8.11 (UKF versus EKF on Artangent Nonlinearity) This example compares the performance of the UKF with that of the EKF for the nonlinear transformation used in Example 8.8, for which we already have a methodology for comparing estimation errors. The values of the mean $\mathrm{E}_x\langle y(x)\rangle$ and variance

TABLE 8.3 Results from Example 8.10: Symmetric Unscented Transform with Alternative Choleksy Factors

Nonlinear Measurement Problem

$$\hat{x} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \; P_{xx} = \begin{bmatrix} 84 & -64 & -32 & 16 \\ -64 & 84 & 16 & -32 \\ -32 & 16 & 84 & -64 \\ 16 & -32 & -64 & 84 \end{bmatrix}, \; z = h(x) = \begin{bmatrix} x_2 x_3 \\ x_3 x_1 \\ x_1 x_2 \end{bmatrix}, \; R = 0$$

Solution by Scaled Unscented Transform with $\alpha = 1, \; \beta = 2, \; \kappa = 2$

Cholesky Factor	Upper Triangular	Lower Triangular	Eigen vector
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 Extended Kalman Filter Approximation

$$\hat{z} \approx h(\hat{x}) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \; H \approx \left.\frac{\partial h}{\partial x}\right|_{x=\hat{x}} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \; P_{zz} \approx H P_{xx} H^{\mathrm{T}} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Results generated by the MATLAB m-file `UTscaledDemo.m`, using the MATLAB function `UTscaled`.

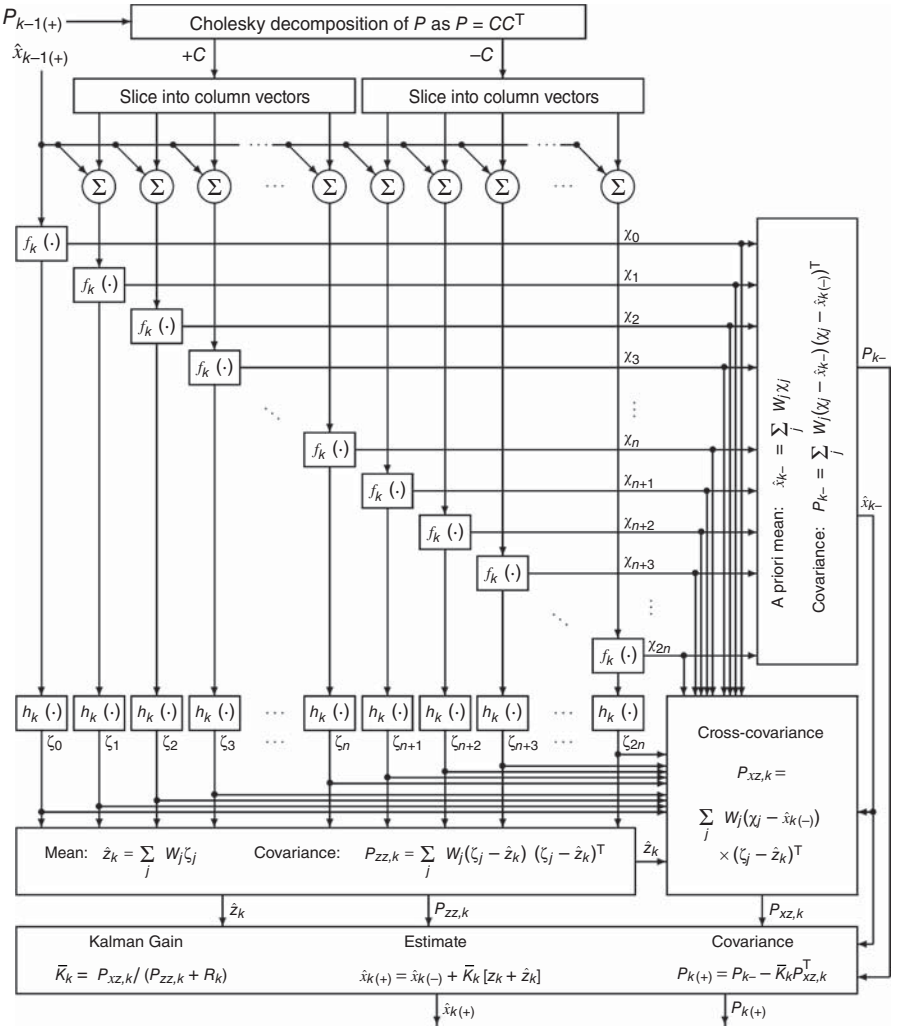


Figure 8.19 Data flow for symmetric UKF with weighting.

$E_x\langle[y(x) - E_\chi\langle y(\chi)\rangle]^2\rangle$ from the EKF and UKF are compared to the value obtained by numerical integration of the expected values involved, assuming the initial probability is Gaussian with mean and variance

$$\hat{x}_{k-1\ (+)} = 0$$
$$P_{k-1\ (+)} = 1.$$

EKF Implementation: The nonlinear temporal state transition

$$x_k = \arctan(a\, x_{k-1})$$

would be modeled in an EKF as

$$\begin{aligned}\hat{x}_{k\,(-)} &= \arctan(a\,\hat{x}_{k-1\,(+)}) \\ &= 0 \\ P_{k\,(-)} &= \left[\left.\frac{\partial \arctan\,(ax)}{\partial x}\right|_{x=\hat{x}_{k-1\,(+)}}\right]^2 P_{k-1\,(+)} \\ &= a^2.\end{aligned}$$

UKF Implementation: The equivalent update for the UKF implementation would be

$$\begin{aligned}\hat{x}_{k\,(-)} &= \arctan(a\,\hat{x}_{k-1\,(+)}) \\ &= 0 \\ P_{k\,(-)} &= [\arctan(a\sqrt{P_{k-1\,(+)}})]^2 \\ &= [\arctan(a)]^2.\end{aligned}$$

Numerical Integration: The mean of the transformed variate will be zero because the arctangent is an odd function. Both the EKF and UKF transform the mean correctly. As a check of how they do on variances, numerical integration of the variance uses a "democratic sampling" of the univariate distribution, in which the real line is divided into N conterminous segments with each segment representing one Nth of the cumulative Gaussian probability. Each segment is represented by its median, which is the mid-point of its cumulative probability. Then the expected value of any function $f(x)$ on the real line can be approximated as[10]

$$\underset{x}{\mathrm{E}}\langle f(x)\rangle = \int_{-\infty}^{+\infty} f(x)\; p(x)\; dx \tag{8.105}$$

$$\approx \frac{1}{N}\sum_{n=1}^{N} f(x_n), \tag{8.106}$$

where the x_n are the medians of the segments and the function

$$f(x) = \arctan^2(ax), \tag{8.107}$$

with scaling parameter values $a > 0$ varied to show how scaling influences the variance of the transformed distribution.

[10]The summation formula can be implemented as a dot-product in MATLAB.

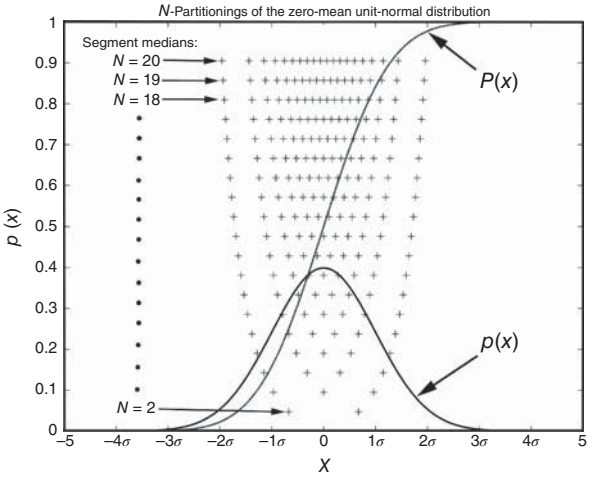


Figure 8.20 Equiprobability partitionings of a Gaussian distribution.

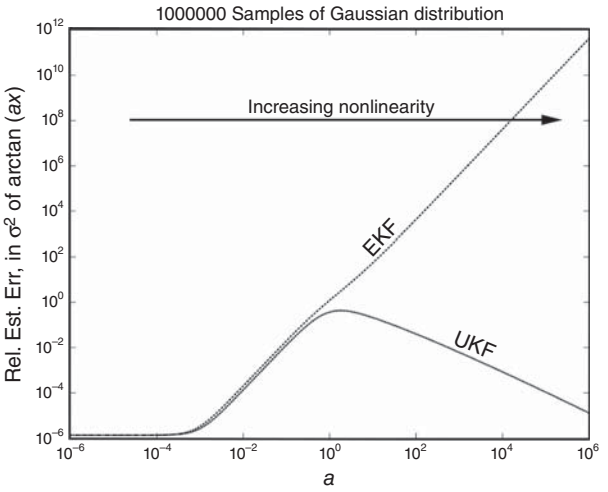


Figure 8.21 Relative error in variance estimates.

The medians of the segments are plotted as "+" symbols in Figure 8.20 for $2 \leq N \leq 20$. A value of $N = 10^6$ was used in the numerical integrations, in which case the sample values of x range from about $-4.9\ \sigma$ to $+4.9\ \sigma$.

Results Results are shown in Figure 8.21, in which values of the numerically integrated variances are compared to the EKF and UKF approximations as a ranges over 12 orders of magnitude. On the plots, the UKF values are denoted by "UKF" and the EKF values by "EKF."

This is a plot of the relative error in the EKF and UKF estimates of variance, in which the relative error in estimating the transformed variance for the EKF approach grows without bound as a increases. The worst relative error of the UKF approximation, on the other hand, gets progressively better when $a \gg 1$.

The result clearly shows—for this particular example—the superiority of the unscented transformation of the variance estimate over that of the EKF. However, the peak relative error in the UKF solution for this problem is still unacceptably high—around 43%.

A tutorial-level introduction to the UKF (with MATLAB implementations) by Yi Cao can be found at

```
http://www.mathworks.com/matlabcentral/fileexchange/18217-learning-
the-unscented-kalman-filter/content/ukf.m.
```

8.5.3 Unscented *sigmaRho* Filtering

The *sigmaRho* filter (described in Section 7.6) is well suited for the UT. It uses a factorization of the Kalman filter covariance matrix P as

$$P = D_\sigma C_\rho D_\sigma \tag{8.108}$$

$$D_\sigma = \text{diag}[\sigma_1,\ \sigma_2,\ \sigma_3,\ \ldots\ ,\ \sigma_n] \tag{8.109}$$

$$= \begin{bmatrix} \sigma_1 & 0 & 0 & \cdots & 0 \\ 0 & \sigma_2 & 0 & \cdots & 0 \\ 0 & 0 & \sigma_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \sigma_n \end{bmatrix} \tag{8.110}$$

$$C_\rho = \begin{bmatrix} 1 & \rho_{12} & \rho_{13} & \cdots & \rho_{1n} \\ \rho_{21} & 1 & \rho_{23} & \cdots & \rho_{2n} \\ \rho_{31} & \rho_{32} & 1 & \cdots & \rho_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho_{n1} & \rho_{n2} & \rho_{n3} & \cdots & 1 \end{bmatrix}, \tag{8.111}$$

where the σ_i are the standard deviations of the error distribution and the ρ_{ij} are the correlation coefficients.

Therefore, if A_ρ is any solution of

$$A_\rho A_\rho^{\mathrm{T}} = C_\rho, \tag{8.112}$$

then $D_\sigma A_\rho$ is a "square root" (generalized Cholesky factor[11]) of P. The columns of $D_\sigma A_\rho$ can then be put through any nonlinear dynamic function f and measurement function h to propagate the covariance matrix and compute the Kalman gain.

[11]Cholesky factors are defined by the Cholesky factoring algorithm, the output of which is a lower triangular matrix L. A "generalized" Cholesky factor has the form $C = LU$, where U is orthogonal.

8.6 TRULY NONLINEAR ESTIMATION

The early history of development of stochastic systems theory has been described in Section 4.2.1. Up to the time the Kalman filter was published, major contributors included Max Planck (1858–1947), Paul Langevin (1872–1946), Marian Smoluchowski (1872–1917), Albert Einstein (1879–1955), Adriaan Fokker (1887–1972), Andrey Kolmogorov (1903–1987), and Ruslan Stratonovich (1930–1997). The Kalman filter was the first exact and practical solution to estimation problem, but it assumed linear dynamic and measurement models.

Few practical nonlinear solutions have emerged since that time, but application areas have included military surveillance, economic system modeling, and market modeling—disciplines that have not been overly biased toward open publication. The following are two results that have appeared in the open literature.

8.6.1 The Beneš Filter

This is perhaps the most successful of the higher order derivations, in that it has an exact solution. It is one of the many discoveries of Czech-American mathematician Václav Beneš[12] [3], this one a finite-dimensional representation of a class of filters using linear measurements with additive noise and nonlinear stochastic dynamics satisfying certain regularity conditions.

It is equivalent to the Kalman filter when the dynamic model is linear.

The Beneš filter model has the form

$$\frac{d}{dt}x(t) = f(x) + Gw(t) \text{ (nonlinear time-invariant)} \tag{8.113}$$

$$z_k = H_k x_k + v_k \text{ (linear discrete)} \tag{8.114}$$

$$\underset{w}{\mathrm{E}}\langle w(t)w^{\mathrm{T}}(t)\rangle = Q \text{ (time-invariant and nonsingular)} \tag{8.115}$$

$$v_k \in \mathcal{N}(0, R), \tag{8.116}$$

plus additional constraints on the unconditioned probability density function of the state vector.

Derivations of the Beneš filter implementation formulas can be found in References 3 and 27, and comparative studies can be found in References 27-29. These tend to show that computational requirements for real-time implementation of the Beneš filter can easily be overwhelming.

We mention the Beneš filter here because it provides an example of a finite-dimensional nonlinear stochastic estimator, although it did not compare well with the EKF in some studies [29]. It has probably been used more in financial market modeling than as a nonlinear extension of Kalman filtering. Its asymptotic performance has been analyzed by Ocone [30]. Farina et al. [29] have compared its

[12]The Czech "š" in proper names is not uncommonly transliterated as "s," although it sounds more like the English "sh."

performance to that of the EKF, linearized filters, and particle filters on a specific problem with known solution. Interested readers are referred to the expositions by Beneš [3,35], Daum [31], Bain and Crisan [27], and the references therein.

8.6.2 Surveillance Solution of Richardson and Marsh

This could be viewed as a nonlinear alternative to multilocal linearization for solving the detection problem. It is what would be called a *grid-based* method for numerical solution of the nonlinear detection and estimation problem. It solves an approximation of the conditioned Fokker–Planck equation on a regular grid in state space.

The following discussion is about an example of an extension of the open-source theory applied to integrated detection and tracking, starting with what is called *the conditioned Fokker–Planck equation* [32], a stochastic differential equation modeling the evolution over time of a state probability density function, given measurements related to the system state.

This particular application is for *surveillance problems*, which include the detection, identification, and tracking of objects within a certain region of state space, where the initial distribution of potential objects to be detected and tracked is according to some random "point process model."

A *point process* is a type of random process for modeling events or objects that are distributed over time and/or space, such as the arrivals[13] 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. Results for these applications tend to be classified, so that there is scant information about capabilities in the open literature. The one statistical surveillance method mentioned here is in the open literature [33], but it is not known how this might compare with classified methods.

A unified approach of this sort, combining detection and tracking into one optimal estimation method, was derived by the late John M. Richardson (1918–1996) and specialized to several applications [33]. 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 moments. Using object densities in place of single-object probabilities makes the approach applicable to the multiobject surveillance problem. The result is a single partial differential equation approximating the evolution over time of the density of objects, based on the observations. The resulting simplified model was programmed into a numerical solution by Richardson and Marsh [33] and applied to a few simple test problems. Results show some promise as a unified solution to the difficult problem of detecting dynamic objects whose initial states are represented by a point process.

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

This would be classified as a "grid-based" approach, in that its output is a set of probability densities evaluated on a grid covering the searchable surveillance space.

8.7 SUMMARY

8.7.1 Main Points

1. As a rule, nonlinear extensions of the Kalman filter are designed to yield the same result as the Kalman filter when the model functions are linear and to degrade gradually when the model becomes increasing less linear.
2. Affine (linear plus offset) filter models are equivalent to models with nonzero-mean noise sources, both can be accommodated with very minor changes in the Kalman filter equations and both result in exact solutions.
3. The EKF has enjoyed a long career with "ever-so-slightly" nonlinear applications and with considerable success. For many problems with acceptable linear approximation using analytic partial derivatives, it is still the most efficient implementation.
4. The UKF has about the same computation requirements as the EKF using numerical partial derivatives and provides an additional margin of robustness against nonlinear effects.
5. The Beneš filter is the only nonlinear extension of the Kalman filter (besides the affine Kalman filter) that is not an approximation, but its performance compared to the EKF or UKF on some applications has not been promising.
6. Any assessment of the relative efficacies of the various nonlinear approximations will be highly dependent on the particulars of the application(s) assumed. Such assessments are best used for picking which approximation to choose for your particular application.

8.7.2 Limitations of Nonlinear Approximation

Truly nonlinear estimation, in general, has reasonably good models from the standpoint of representing the evolution over time of probability distributions. However, their computational complexity tends to rule them out for the sort of real-time applications for which Kalman filtering has been so useful [37].

Except for the affine and Beneš filters, all existing nonlinear extensions of the Kalman filter are approximations of one sort or another.

The mean and covariance matrix are sufficient statistics for minimum mean-squared error estimation only if the measurements and dynamics are linear (or affine) functions of the state vector.

Otherwise, the mean and covariance matrix will be corrupted by the higher order moments of the distribution which are not propagated in the linearized, extended, or sample-and-propagate filters. Approximation methods attempt to minimize this

corruption, but they cannot guarantee to reduce it to acceptable levels for all nonlinear models. The remarkable thing is that sample-and-propagate methods have performed as well as they have, given the inherent insufficiency of information for propagating means and covariances.

The effects of linear approximation errors can be approximated for the EKF. Comparable assessments for sample-and-propagate approximations have been done assuming Gaussian distributions. However, nonlinear transformations do not preserve Gaussianity.

PROBLEMS

8.1 A scalar stochastic sequence x_k is given by

$$\begin{aligned} x_k &= -0.1x_{k-1} + \cos x_{k-1} + w_{k-1}, \quad z_k = x_k^2 + v_k, \\ \mathrm{E}\langle w_k \rangle &= 0 = \mathrm{E}\langle v_k \rangle, \quad \mathrm{cov} w_k = \Delta(k_2 - k_1), \quad \mathrm{cov} v_k = 0.5\Delta(k_2 - k_1), \\ \mathrm{E}\langle x_0 \rangle &= 0, \quad P_0 = 1, \quad x_k^{\mathrm{nom}} = 1. \end{aligned}$$

Determine the linearized and extended Kalman estimator equations.

8.2 A scalar stochastic process $x(t)$ is given by

$$\begin{aligned} \dot{x}(t) &= -0.5x^2(t) + w(t), \\ z(t) &= x^3(t) + v(t), \\ \mathrm{E}\langle w(t) \rangle &= \mathrm{E}\langle v(t) \rangle = 0 \\ \mathrm{cov} w_t &= \delta(t_1 - t_2), \quad \mathrm{cov} v(t) = 0.5\delta(t_1 - t_2), \\ \mathrm{E}\langle x_0 \rangle &= 0, \quad P_0 = 1, \quad x^{\mathrm{nom}} = 1. \end{aligned}$$

Determine the linearized and extended Kalman estimator equations.

8.3 **(a)** Verify the results of Example 8.6 (IEKF).

(b) Estimate the states from a noisy data.

(c) Compare the results of linearized Kalman filter and EKF.
Assume that the plant noise is normally distributed with mean 0 and covariance 0.2 and measurement noise is normally distributed with mean 0 and covariance 0.001.

8.4 Derive the linearized and EKF equations for the following equations:

$$x_k = f_{k-1}(x_{k-1}) + Gw_{k-1}, \quad z_k = h_k(x_k) + v_k.$$

8.5 Given the following plant and measurement model for a scalar dynamic system:

$$\begin{aligned} \dot{x}(t) &= ax(t) + w(t), \quad z(t) = x(t) + v(t), \\ w(t) &\sim \mathcal{N}(0,1), \quad v(t) \sim \mathcal{N}(0,2) \\ \mathrm{E}\langle x(0)\rangle &= 1 \\ \mathrm{E}\langle w(t)v(t)\rangle &= 0 \\ P(0) &= 2, \end{aligned}$$

assume an unknown constant parameter a and derive an estimator for a, given $z(t)$.

8.6 Let $\mathbf{r}$ represent the position vector to a magnet with dipole moment vector $\mathbf{m}$. The magnetic field vector $\mathbf{H}$ at the origin of the coordinate system in which $\mathbf{r}$ is measured is given by the formula

$$\mathbf{B} = \frac{\mu_0}{4\pi|\mathbf{r}|^5}\{3\mathbf{r}\mathbf{r}^{\mathrm{T}} - |\mathbf{r}|^2\mathbf{I}\}\mathbf{m} \tag{8.117}$$

in SI units.

(a) Derive the measurement sensitivity matrix for $\mathbf{H}$ 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?

8.7 Generate the error covariance results for the plant and measurement models given in Example 8.4 with the given values of process and measurement noise covariance and initial state estimation error covariance.

8.8 This problem is taken from Reference 34. The equations of motion for the space vehicle are given as

$$\ddot{r} - r\dot{\theta}^2 + \frac{k}{r^2} = w_r(t), \quad r\ddot{\theta} + 2\dot{r}\dot{\theta} = w_\theta(t),$$

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

$$z(t) = \begin{bmatrix} \sin^{-1}\frac{R_e}{r} \\ \alpha_0 - \theta \end{bmatrix},$$

where R_e is the earth radius and α_0 is a constant.
Linearize these equations about $r_{\mathrm{nom}} = R_0$ and $\theta_{\mathrm{nom}} = w_0 t$.

8.9 Let the 3×3 covariance matrix[14]

$$P = \begin{bmatrix} 11 & 7 & -9 \\ 7 & 11 & -9 \\ -9 & -9 & 27 \end{bmatrix}. \tag{8.118}$$

(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 $C^{\mathrm{T}}C = P$, in which case C^{T} is a lower triangular Cholesky factor.

(b) Use the MATLAB function `utchol` on the Wiley web site 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 $P = U\Lambda V^{\mathrm{T}}$, where $V = U$.) The square root of P is then $S_P = U\Lambda^{1/2}U^{\mathrm{T}}$, 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?

8.10 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 $\hat{x} = 0$, and do not forget to multiply your Cholesky factors by $\sqrt{n} = \sqrt{3}$ to obtain the properly scaled sigma points.

8.11 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

$$\vec{f}\left(\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}\right) = \begin{bmatrix} x_2 x_3 \\ x_3 x_1 \\ x_1 x_2 \end{bmatrix}.$$

8.12 Compare the resulting means from the previous problem to the exact mean

$$\mathrm{E}\left\langle \begin{bmatrix} x_2 x_3 \\ x_3 x_1 \\ x_1 x_2 \end{bmatrix} \right\rangle = \begin{bmatrix} p_{2,3}\left(1 - p_{2,3}^2/p_{2,2}/p_{3,3}\right)^{-3/2} \\ p_{3,1}(1 - p_{3,1}^2/p_{3,3}/p_{1,1})^{-3/2} \\ p_{1,2}(1 - p_{1,2}^2/p_{1,1}/p_{2,2})^{-3/2} \end{bmatrix}$$

for Gaussian initial distributions with zero mean and covariance P. Use the value of P from Equation 8.118 in this calculation.

[14]The value of P in Equation 8.118 was used to generate the equiprobability ellipsoid shown in Figure 8.12.

8.13 Rework Example 8.4 using the symmetric UKF implementation with $2n + 1$ 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 $\dot{Q} = FQF^{\mathrm{T}} + Q_t$ with $Q(t_{k-1}) = 0$ and $F = \frac{\partial \dot{x}}{\partial x}$.

8.14 Is a symmetric square root of a symmetric positive-definite matrix also a Cholesky factor? Explain your answer.

REFERENCES

[1] 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.

[2] 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, International Federation on Automatic Control, Laxenburg, Austria, pp. 140–144, 1991.

[3] V. E. Beneš, "Exact finite-dimensional filters with certain diffusion non-linear drift," *Stochastics*, Vol. 5, pp. 65–92, 1981.

[4] 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.

[5] 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.

[6] A. P. Andrews, U.S. Patent No. 5381095, "Method of estimating location and orientation of magnetic dipoles using extended Kalman filtering and Schweppe likelihood ratio detection," 1995.

[7] A. P. Andrews, "The accuracy of navigation using magnetic dipole beacons," *Navigation, The Journal of the Institute of Navigation*, Vol. 38, pp. 369–397, 1991–1992.

[8] F. C. Schweppe, "Evaluation of likelihood functions for Gaussian signals," *IEEE Transactions on Information Theory*, Vol. IT-11, pp. 61–70, 1965.

[9] F. L. Markley. "Attitude error representations for Kalman Filtering," *Journal of Guidance, Control, and Dynamics*, Vol. 26, No. 2, pp. 311–317, 2003.

[10] S. M. Ulam, *Adventures of a Mathematician*, University of California Press, Berkeley, CA, 1991.

[11] N. Metropolis, "The beginning of the Monte Carlo method," *Los Alamos Science*, No. 15, Special Issue: Stanislaw Ulam (1909–1984), pp. 125–130, 1987.

[12] 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.

[13] S. J. Julier, J. K. Uhlmann, and H. F. Durrenat-Whyte, "A new approach to filtering nonlinear systems," in *Proceedings of the 1995 American Control Conference*, Seattle, WA, pp. 1628–1632, 1995.

[14] 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, Oxford, 1996.

[15] 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, pp. 182–193, 1997.

[16] 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.

[17] 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*, Anchorage, Alaska, Vol. 2, pp. 887–892, 2002.

[18] 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.

[19] S. J. Julier, "The spherical simplex unscented transformation," in *Proceedings of the 2003 American Control Conference*, Denver, Colorado, Vol. 3, pp. 2430–2434, 2003.

[20] S. J. Julier and J. K. Uhlmann, "Unscented filtering and nonlinear estimation," *Proceedings of the IEEE*, Vol. 92, No. 3, pp. 401–422, 2004.

[21] S. J. Julier and J. K. Uhlmann, "Corrections to Unscented filtering and nonlinear estimation," *Proceedings of the IEEE*, Vol. 92, No. 12, p. 1958, 2004.

[22] J. J. LaViola, "A comparison of unscented and extended Kalman filtering for estimating quaternion motion," in *Proceedings of the 2003 American Control Conference*, Denver, Colorado, Vol. 3, pp. 2435–2440, June2003.

[23] R. Van der Merwe and E. A. Wan, "The square-root unscented Kalman filter for state and parameter-estimation," in *Proceedings of 2001 IEEE International Conference on Acoustics, Speech, and Signal Processing*, Salt Lake City, Utah, Vol. 6, pp. 3461–3464, 2001.

[24] J. R. Van Zandt, "A more robust unscented transform," *Proceedings of SPIE Conference on Signal and Data Processing of Small Targets*, San Diego, CA, 2001.

[25] T. Lefebvre, H. Bruyninckx, and J. De Schutter, *"The linear regression Kalman filter,"* in *Nonlinear Kalman Filtering for Force-Controlled Robot Tasks*, *Springer Tracts in Advanced Robotics*, Springer-Verlag, Berlin, Vol. 19, pp. 205–210, 2005.

[26] T. Lefebvre, H. Bruyninckx, and J. De Schutter, "Comment on A new method for the nonlinear transformation of means and covariances in filters and estimators," *IEEE Transactions on Automatic Control*, Vol. 47, No. 8, pp. 1406–1408, 2002.

[27] A. Bain and D. Crisan, *Fundamentals of Stochastic Filtering*, *Stochastic Modeling and Applied Probability Series*, No. 60, Springer, New York, 2009.

[28] L. V. Bagaschi, "A comparative study of nonlinear tracking algorithms," Doctorate dissertation, Swiss Federation of Technology (ETH), Zurich, 1991.

[29] A. Farina, D. Benvenuti, and B. Ristic, "A comparative study of the Benes filtering problem," *Signal Processing*, Vol. 82, No. 2, pp. 133–147, 2002.

[30] D. L. Ocone, "Asymptotic stability of Beneš filters," *Stochastic Analysis and Applications*, Vol. 17, No. 6, pp. 1053–1074, 1999.

[31] F. E. Daum, "Exact finite-dimensional nonlinear filters," *IEEE Transactions on Automatic Control*, Vol. AC-31, No. 7, pp. 616–622, 1986.

[32] H. Risken and T. Frank, *The Fokker–Planck equation: methods of solution and applications*, 2nd ed., Springer, Berlin, 1989.

[33] J. M. Richardson and K. A. Marsh, *"Point process theory and the surveillance of many objects,"* in *Maximum Entropy and Bayesian Methods*, Seattle, Kluwer Academic Publishers, Norwell, MA, pp. 213–220, 1991.

[34] H. W. Sorenson, *"Kalman filtering techniques,"* in *Advances in Control Systems* Vol. 3 (C. T. Leondes, ed.), Academic Press, New York, pp. 219–292, 1966.

[35] V. Beneš and R. J. Elliott, "Finite-dimensional solutions of a modified Zakai equation," *Mathematics of Control, Signals, and Systems*, 9, pp. 341–351, 1996.

[36] A. Budhirajaa, L. Chenb, and C. Leea, "A survey of numerical methods for nonlinear filtering problems," *Physica D: Nonlinear Phenomena*, Vol. 230, No. 1 and 2, pp. 27–36, 2007.

[37] A. Doucet, N. de Freitas, and N. Gordon, *Sequential Monte Carlo Methods in Practice*, Springer, New York, 2001.

[38] R. Eckhardt, "Stan Ulam, John von Neumann, and the Monte Carlo Method," *Los Alamos Science*, No. 15, Special Issue: Stanislaw Ulam (1909–1984), pp. 131–141, 1987.

[39] B. Ristic, S. Arulampalam, and N. Gordon, *Beyond the Kalman Filter: Particle Filters for Tracking Applications*, Artech, Boston, MA, 2004.

9

PRACTICAL CONSIDERATIONS

> The classical (more accurately, old-fashioned) view is that a mathematical problem is solved if the solution is expressed by a formula. It is not a trivial matter, however, to substitute numbers in a formula[1].

9.1 CHAPTER FOCUS

The discussion turns now to what might be called *Kalman filter engineering*, which is that body of applicable knowledge that has evolved through practical experience in the use and misuse of the Kalman filter. The material of 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.

9.1.1 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.

[1]R. E. Kalman and R. S. Bucy, "New results in linear filtering and prediction theory," Journal of Basic Engineering, Series D, Vol. 83, pp. 95–108, 1961.

Kalman Filtering: Theory and Practice Using MATLAB®, Fourth Edition.
Mohinder S. Grewal and Angus P. Andrews.
© 2015 John Wiley & Sons, Inc. Published 2015 by John Wiley & Sons, Inc.

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 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. *Offline Processing to Reduce Online 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*. It is a rather simple check for symptoms of mismodeling.

9.2 DIAGNOSTIC STATISTICS AND HEURISTICS

This is a collection of methods that have been found useful in understanding the observed behavior of Kalman filters and for detecting and correcting anomalous behavior.

9.2.1 Innovations Analysis

Innovations[2] are the differences between observed and predicted measurements,

$$\nu_k \stackrel{\text{def}}{=} z_k - H_k \hat{x}_{k(-)}. \tag{9.1}$$

[2]Kailath [1] introduced the notation ν (Greek letter "nu") for innovations, because they represent "what is new" in the measurement.

Innovations are the carotid artery of a Kalman filter. They provide an easily accessible point for monitoring vital health status without disrupting normal operations, and the statistical and temporal properties of its pulses can tell us much about what might be right or wrong with a Kalman filter implementation.

9.2.1.1 Properties of Innovations If the Kalman filter is properly modeled for its task, its innovations

1. Have zero mean

$$\underset{k}{\mathrm{E}}\langle \nu_k \rangle = 0; \tag{9.2}$$

2. Are white (i.e., uncorrelated in time)

$$\underset{k \neq j}{\mathrm{E}} \langle \nu_k \nu_j^{\mathrm{T}} \rangle = 0; \tag{9.3}$$

3. Have known covariance

$$P_{\nu\nu k} \overset{\text{def}}{=} \mathrm{E}_k \langle \nu_k \nu_k^{\mathrm{T}} \rangle \tag{9.4}$$

$$= H_k P_{k(-)} H_k^{\mathrm{T}} + R_k; \tag{9.5}$$

4. Have known information matrix

$$Y_{\nu\nu k} \overset{\text{def}}{=} P_{\nu\nu k}^{-1} \tag{9.6}$$

$$= [H_k P_{k(-)} H_k^{\mathrm{T}} + R_k]^{-1}, \tag{9.7}$$

   which is a partial result in the computation of the Kalman gain;

5. Have a known mean value for the information quadratic form

$$\underset{k}{\mathrm{E}} \langle \nu_k Y_{\nu\nu k} \nu_k^{\mathrm{T}} \rangle = \ell, \tag{9.8}$$

   the dimension of the measurement vector z_k;

6. And—if all the error sources are Gaussian—the information quadratic forms have a chi-squared distribution with ℓ degrees of freedom

$$\{ \nu_k Y_{\nu\nu k} \nu_k^{\mathrm{T}} \} \in \chi_\ell^2. \tag{9.9}$$

Likely causes for anomalous values for these innovations statistics are addressed in the following subsections.

Isolating the cause often comes down to deciding whether the source is the sensor (z_k), the dynamic system model ($H_k \hat{x}_{k(-)}$), or exogenous sources. In either case, it is cause for reexamination of the entire Kalman filtering model and for questioning whether the cause could be some exogenous source such as power supply noise,

mechanical vibrations induced by environmental conditions, or diurnal variations in light levels, temperature, and humidity. Even cyclic variations due to heating, ventilation, and air conditioning systems have been known to create noticeable errors. Analysis of innovations may give some clues about likely source(s).

In all cases, any changes made to correct anomalous behavior detected from innovations diagnostics must be verified by follow-up evaluations of the same diagnostics to confirm the diagnosis.

Example 9.1 (Simple Innovations Test of Mismodeling) As a simple illustration of the general approach, the MATLAB® file `InnovAnalysis.m` in the Wiley web site 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 two. 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 root-mean-square (RMS) differences between the simulated state variable and the estimates from the nine Kalman filters.

The results of a Gaussian Monte Carlo simulation with a 1000 time steps are summarized in Table 9.1. These do indicate that the correctly modeled Kalman filter has a χ^2 statistic close to $\ell = 1$ 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.

These results do indicate that—for this particular application implementation—the innovations mean is a reasonably good indicator of mismodeling of some sort. Changing a parameter by a factor of two may not change the chi-squared mean by a factor of two, but it does change it noticeably.

TABLE 9.1 Example Innovations Analysis of Mismodeled Implementations

Kalman Filter Parameter Values*				Performance		
				A Priori	A Posteriori	χ^2 Mean
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.

9.2.1.2 Diagnosing Means Not-uncommon causes for the means of innovations sequences being nonzero include the following:

1. Nonzero-mean sensor noise, also called *sensor bias error*. If the bias is constant, this can be verified and corrected by recalibration of the sensor(s) involved. Otherwise, sensor bias can be appended to the state vector as an exponentially correlated process or a random walk.
2. Overrated sensor noise, in which the value of R has been set too high. When the state variables are unknown constants, this can cause the Kalman gain to be too small, which causes delayed convergence. This, too can be verified and corrected by recalibration of the sensor(s) or by appending elements of R to the state vector as unknown parameters.
3. Underrated dynamic disturbance noise, which can also lead to lagged convergence. "Tuning" the dynamic disturbance noise covariance is a fairly common remedy for apparent slow convergence, and this is sometimes implemented as a parameter estimation problem. In some cases, however, this may only be masking other mismodeling errors.

9.2.1.3 Autocorrelation Analysis The MATLAB function `xcov` in the Signal Processing Toolbox can be used for computing the autocovariance of innovations, which should be near zero except at zero lag. The zero-lag value should equal P_{vv}, as given by Equation 9.5.

If the zero-lag value is quite different from P_{vv}, there are a number of options, including "tuning" Q or R.

9.2.1.4 Spectrum Analysis This is especially useful for detecting unmodeled resonances, including unmodeled harmonics in sensor noise and any harmonics of the operational environment. The power spectral densities and cross-spectral densities can be computed by taking the fast Fourier transform of the autocovariances from the previous subsection or by computing the spectra and cospectra directly from innovations sequences.

The frequencies of detected harmonics can be useful for deciding whether to look for electronic, vibrational, thermal control, or diurnal variations as likely sources.

If the corrupting harmonics occur only on a single sensor output, that sensor may be suspect. In that case, the sensor noise model can be augmented to include the harmonic noise.

If the same harmonic frequency is spread across multiple outputs, then the inverse sensor transformation $H_k^\dagger$ (Moore–Penrose inverse of H_k) may offer a clue of the system-level source. If, for example, $H_k^\dagger$ maps the problem back to a single state variable, the dynamic model of that state variable can be augmented to include the offending harmonic term.

The center of a harmonic peak identifies the frequency of the missing harmonic model, and the spreading of the harmonic peak may give some indication of the modeled damping factor to be used.

9.2.1.5 Covariance/Information Analysis The analysis described in Section 9.2.1.3 results in a statistic that should resemble $P_{\nu\nu}$, which is computed as a partial result in the calculation of Kalman gain.

If not, then either R_k or $H_k P_{k(-)} H_k^{\mathrm{T}}$ may be blamed and either can be tuned accordingly.

9.2.1.6 Chi-Squared Means The mean value of the innovations-norm sequence

$$\{\nu_k Y_{\nu\nu k} \nu_k^{\mathrm{T}}\}$$

should equal the dimension ℓ of the measurement vector z_k.

If it is larger, then either R_k or $H_k P_{k(-)} H_k^{\mathrm{T}}$ may be blamed, and either can be tuned accordingly.

9.2.1.7 Chi-Squared Distributions The covariance of a chi-squared distribution should be twice its mean

$$\mathrm{E}_k \langle [\nu_k Y_{\nu\nu k} \nu_k^{\mathrm{T}} \ell]^2 \rangle = 2\ell, \tag{9.10}$$

which provides another indicator of whether the distribution of innovations norms is actually chi-squared.

However, the Kalman filter seems to operate well even when the underlying random processes are not necessarily Gaussian, so the discovery that the histograms of an innovations sequence do not look like a chi-squared distribution should perhaps be taken with a grain of salt.

Example 9.2 (Sensor Roundoff Errors) Roundoff errors are ubiquitous in Kalman filter implementations. As a rule, they cannot be monitored because they are not detectable within the precision limits of the implementation—except when the least significant bit (LSB) of a digitized sensor output is bigger than the LSB of the processor (which is not uncommon). In that case, sensor noise can be dominated by roundoff error, which is decidedly non-Gaussian.

In a well-designed digitizer, roundoff errors $\varepsilon_{\mathrm{rnd}}$ tend to be uniformly distributed from $-1/2$ LSB to $+1/2$ LSB. That is $\varepsilon_{\mathrm{rnd}} \in \mathcal{U}([-1/2 \text{ LSB}, +1/2 \text{ LSB}])$, the uniform distribution from $-1/2$ LSB to $+1/2$ LSB. The mean of this distribution is zero and its variance and information will be

$$\begin{aligned}
\sigma_{\mathrm{rnd}}^2 &\stackrel{\mathrm{def}}{=} \mathop{\mathrm{E}}_{\varepsilon_{\mathrm{rnd}}} \langle \varepsilon_{\mathrm{rnd}}^2 \rangle \\
&= \frac{1}{\mathrm{LSB}} \int_{-1/2 \text{ LSB}}^{+1/2 \text{ LSB}} \varepsilon^2 \, d\varepsilon \\
&= \frac{1}{\mathrm{LSB}} \left[\frac{\varepsilon^3}{3} \right]_{-1/2 \text{ LSB}}^{+1/2 \text{ LSB}} \\
&= \mathrm{LSB}^2/12
\end{aligned}$$

$$Y_{\text{rnd}} = 12/\text{LSB}^2,$$

respectively. Sensor roundoff errors could then dominate the innovations

$$\begin{aligned} &\text{if } R \approx \text{LSB}^2/12 \\ &\text{and } R \gg HPH^{\text{T}} \\ &\text{then } \nu_k \in \mathcal{U}([-1/2 \text{ LSB}, +1/2 \text{ LSB}]) \\ &\text{and } Y_{\nu\nu} \approx Y_{\text{rnd}} \\ &\qquad = 12/\text{LSB}^2. \end{aligned}$$

The analog of the χ^2 statistic in this case is the information-normalized square of the innovations $\nu = \varepsilon_{\text{rnd}}$

$$\begin{aligned} |\varepsilon_{\text{rnd}}|_Y^2 &\stackrel{\text{def}}{=} \varepsilon_{\text{rnd}}^2 \times Y_{\text{rnd}} \\ &= \varepsilon_{\text{rnd}}^2 \frac{12}{\text{LSB}^2}, \end{aligned}$$

the mean of which is 1 (one), the same as the mean of the χ^2 distribution with one degree of freedom, χ_1^2. Therefore, sensor roundoff errors—even though they are uniformly distributed—will pass the zero-mean and chi-squared mean tests. However, the distribution of this statistic is not chi-squared.

Using Equation 3.127, the distribution of $y = |\varepsilon_{\text{rnd}}|_Y^2$ can be derived as

$$p(y) = \begin{cases} 0, & y < 0 \\ \frac{1}{2\sqrt{3y}}, & 0 \le y \le 3, \\ 0, & y > 3 \end{cases}$$

as plotted in Figure 9.1. Figure 9.1(a) shows the roundoff error distribution compared to the Gaussian distribution with the same mean and variance. Figure 9.1 (b) is a semi-log plot showing the relevant squared roundoff distribution compared to the χ_1^2 distribution.

These results would indicate that an empirical distribution based on observed innovations may be preferable to the chi-square distribution for assessing whether an observed innovation is acceptable.

9.2.1.8 Real-Time Monitoring Because numerical values of the innovations ν_k and their alleged information matrix $Y_{\nu\nu k}$ are available "for free" within the Kalman filter implementation, the marginal computational cost of monitoring innovations statistics is relatively low. The expected payoffs for such monitoring will depend on attributes of the application. In all cases, however, such monitoring will generally require

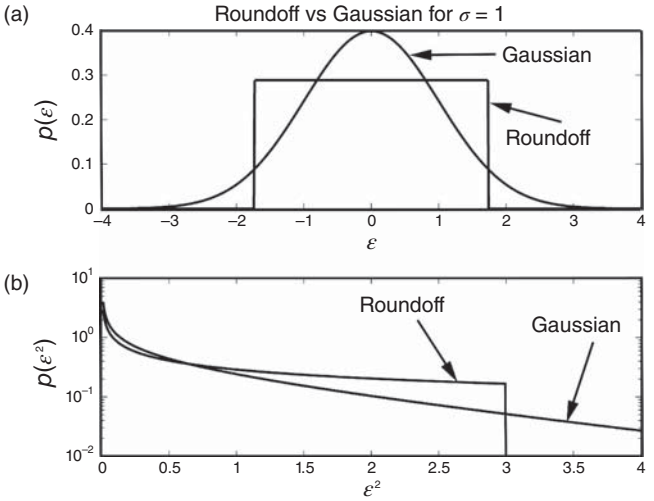


Figure 9.1 Distributions of roundoff errors.

1. Selection of which statistics to monitor, based on relative efficacy in separating anomalous events from normal events.
2. Determination of threshold levels for determining which events are out of tolerance. This may involve some analysis related to the relative costs of false negatives and false positives.
3. Exception-handling software for making decisions about what needs to be done and carrying out the necessary actions.

It is common practice in some applications to save the monitored statistics for offline analysis, as well—particularly during early testing and evaluation.

Monitoring of this sort may also be done for spotting performance degradation trends, based on long-term statistical values.

9.2.2 Convergence and Divergence

9.2.2.1 Some Definitions 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[3] if, for every $\varepsilon > 0$, for some integer n, for all $k, \ell > n$, $\|\eta_k - \eta_\ell\| < \varepsilon$.

[3]Such sequences are called *Cauchy sequences*, after Augustin Louis Cauchy (1789–1857).

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*.

9.2.2.2 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.

9.2.2.3 Variables Subject to 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;
$\mathrm{E}\langle x_k x_k^{\mathrm{T}}\rangle$, 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 estimation error.

9.2.3 Covariance Analysis

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 7 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.

9.2.4 Testing for Unpredictable Behavior

Not all filter divergence is predictable from the Riccati equation solution. Sometimes the actual performance does not agree with theoretical performance.

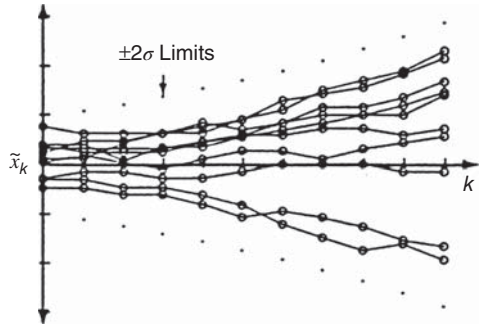


Figure 9.2 Dispersion of multiple runs.

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 9.2, 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,

$$\frac{1}{N}\sum_{i=1}^{N}[\hat{x}_{[i]}(t_k) - x(t_k)] \approx \mathop{\mathrm{E}}_{i}\langle \hat{x}_{[i]}(t_k) - x(t_k)\rangle = 0, \tag{9.11}$$

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.

9.2.4.1 Convergence of Suboptimal Filters In the suboptimal filters discussed in Section 9.5, the estimates can be biased. Therefore, in the analysis of suboptimal filters, the behavior of $P(t)$ is not sufficient to define convergence. A suboptimal filter is said to converge if the covariance matrices converge,

$$\lim_{t\to\infty}[\mathrm{trace}(P_{\text{sub-opt}} - P_{\text{opt}})] = 0, \tag{9.12}$$

and the asymptotic estimation error is unbiased,

$$\lim_{t\to\infty}\mathrm{E}[\tilde{x}(t)] = 0. \tag{9.13}$$

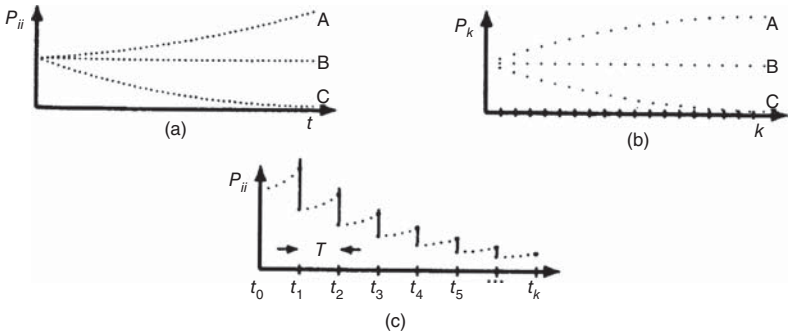


Figure 9.3 Asymptotic behaviors of estimation uncertainties. (a) Continuous time, (b) discrete time, and (c) discrete measurement with continuous dynamics.

Example 9.3 Some typical behavior patterns of suboptimal filter convergence are depicted by the plots of $P(t)$ in Figure 9.3(a), and characteristics of systems with these symptoms are given here as examples.

Case A: Let a scalar continuous system equation be given by

$$\dot{x}(t) = Fx(t), \qquad F > 0, \tag{9.14}$$

in which the system is unstable, or

$$\dot{x}(t) = Fx(t) + w(t) \tag{9.15}$$

in which the system has driving noise and is unstable.

Case B: The system has constant steady-state uncertainty:

$$\lim_{t\to\infty} \dot{P}(t) = 0. \tag{9.16}$$

Case C: The system is stable and has no driving noise:

$$\dot{x}(t) = -Fx(t), \qquad F > 0. \tag{9.17}$$

Example 9.4 (Behaviors of Discrete-Time Systems) Plots of P_k are shown in Figure 9.3(b) for the following system characteristics:

Case A: Effects of dynamic disturbance 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 dynamic disturbance noise and measurement noise are small relative to $P_0(t)$.

Example 9.5 (Continuous System with Discrete Measurements) A scalar example of a behavior pattern of the covariance propagation equation ($P_k(-), \dot{P}(t)$) and covariance update equation $P_k(+)$,

$$\dot{x}(t) = Fx(t) + w(t), F < 0,$$
$$z(t) = x(t) + v(t),$$

is shown in Figure 9.3(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 9.3(c).

9.2.4.2 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 9.6 The "natural behavior" for P in some cases is for

$$\lim_{t\to\infty} P(t) = P_\infty \text{ (a constant).} \tag{9.18}$$

For example,

$$\left.\begin{array}{ll} \dot{x} = w, & \text{cov}(w) = Q \\ z = x + v, & \text{cov}(v) = R \end{array}\right) \Rightarrow \begin{array}{l} F = 0 \\ G = H = 1 \end{array} \quad \text{in} \quad \begin{array}{l} \dot{x} = Fx + Gw, \\ z = Hx + v. \end{array} \tag{9.19}$$

Applying the continuous Kalman filter equations from Chapter 5, 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

$$P(t) = \alpha \left(\frac{P_0 \cosh(\beta t) + \alpha \sinh(\beta t)}{P_0 \sinh(\beta t) + \alpha \cosh(\beta t)} \right), \tag{9.20}$$

where

$$\alpha = \sqrt{RQ}, \quad \beta = \sqrt{Q/R}. \tag{9.21}$$

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 9.4(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) and not in software.

Example 9.7 (Divergence Due to "Structural" Unobservability) The filter is said to diverge at infinity if its limit is unbounded:

$$\lim_{t\to\infty} P(t) = \infty. \tag{9.22}$$

As an example in which this occurs, consider the system

$$\begin{aligned} \dot{x}_1 &= w, & \operatorname{cov}(w) &= Q, \\ \dot{x}_2 &= 0, & \operatorname{cov}(v) &= R, \\ z &= x_2 + v, \end{aligned} \tag{9.23}$$

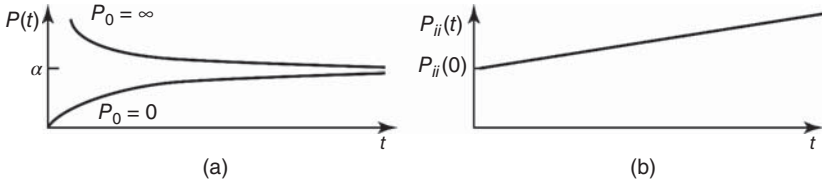


Figure 9.4 Behavior patterns of P. (a) Convergent to finite limit and (b) convergent to infinite limit.

with initial conditions

$$P_0 = \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix} = \begin{bmatrix} P_{11}(0) & 0 \\ 0 & P_{22}(0) \end{bmatrix}. \tag{9.24}$$

The continuous Kalman filter equations

$$\begin{aligned} \dot{P} &= FP + PF^{\mathrm{T}} + GQG^{\mathrm{T}} - \overline{K}R\overline{K}^{\mathrm{T}}, \\ \overline{K} &= PH^{\mathrm{T}}R^{-1} \end{aligned}$$

can be combined to give

$$\dot{P} = FP + PF^{\mathrm{T}} + GQG^{\mathrm{T}} - PH^{\mathrm{T}}R^{-1}HP, \tag{9.25}$$

or

$$\dot{P}_{11} = Q - \frac{p_{12}^2}{R}, \quad \dot{P}_{12} = -\frac{p_{12}p_{22}}{R}, \quad \dot{P}_{22} = -\frac{p_{22}^2}{R}, \tag{9.26}$$

the solution to which is

$$p_{11}(t) = p_{11}(0) + Qt, \quad p_{12}(t) = 0, \quad p_{22}(t) = \frac{p_{22}(0)}{1 + [p_{22}(0)/R]t}, \tag{9.27}$$

as plotted in Figure 9.4(b). The only remedy in this example is to alter or add measurements (sensors) to achieve observability.

Example 9.8 (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

$$0 < \lim_{k \to \infty} P_k < \infty \tag{9.28}$$

is actually bounded away from zero uncertainty.

Parameter Estimation Model for Continuous Time Consider the two-dimensional parameter estimation problem

$$\left.\begin{aligned} \dot{x}_1 = 0, \qquad \dot{x}_2 = 0, P_0 = \begin{bmatrix} \sigma_1^2(0) & 0 \\ 0 & \sigma_2^2(0) \end{bmatrix}, \quad H = \begin{bmatrix} 1 & 1 \end{bmatrix}, \\ z = H \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + v, \qquad \mathrm{cov}(v) = R, \end{aligned}\right\} \tag{9.29}$$

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 only works here 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 5.2):

$$x_k^1 = x_{k-1}^1 \quad (x^1 \text{ is constant}), \tag{9.30}$$

$$x_k^2 = x_{k-1}^2 \quad (x^2 \text{ is constant}), \tag{9.31}$$

$$z_k = \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} x_k^1 \\ x_k^2 \end{bmatrix} + v_k. \tag{9.32}$$

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,

$$z_1 = (x_1^1 + x_1^2), \quad \text{where } v_1 \to 0 \quad \text{and} \quad R = \text{cov}(v_1) \to 0. \tag{9.33}$$

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

$$\overline{K}_1 = \begin{bmatrix} \dfrac{\sigma_1^2(0)}{(\sigma_1^2(0) + \sigma_2^2(0) + R)} \\ \dfrac{\sigma_2^2(0)}{\sigma_1^2(0) + \sigma_2^2(0) + R} \end{bmatrix}. \tag{9.34}$$

The estimation uncertainty covariance matrix can then be shown to be

$$P_{1(+)} = \begin{bmatrix} \dfrac{\sigma_1^2(0)\sigma_2^2(0) + R\sigma_1^2(0)}{\sigma_1^2(0) + \sigma_2^2(0) + R} & \dfrac{-\sigma_1^2(0)\sigma_2^2(0)}{\sigma_1^2(0) + \sigma_2^2(0) + R} \\ \dfrac{-\sigma_1^2(0)\sigma_2^2(0)}{\sigma_1^2(0) + \sigma_2^2(0) + R} & \dfrac{\sigma_1^2(0)\sigma_2^2(0) + R\sigma_2^2(0)}{\sigma_1^2(0) + \sigma_2^2(0) + R} \end{bmatrix} \equiv \begin{bmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{bmatrix}, \tag{9.35}$$

where the *correlation coefficient* (defined in Equation 3.40) is

$$\rho_{12} = \frac{p_{12}}{\sqrt{p_{11}p_{22}}} = \frac{-\sigma_1^2(0)\sigma_2^2(0)}{\sqrt{[\sigma_1^2(0)\sigma_2^2(0) + R\sigma_1^2(0)][\sigma_1^2(0)\sigma_2^2(0) + R\sigma_2^2(0)]}}, \tag{9.36}$$

and the state estimate is

$$\hat{x}_1 = \hat{x}_1(0) + \overline{K}_1[z_1 - H\hat{x}_1(0)] = [I - \overline{K}_1\, H]\hat{x}_1(0) + \overline{K}_1 z_1. \tag{9.37}$$

However, for the *noise-free* case,

$$v_1 = 0, \quad R = \text{cov}(v_1) = 0,$$

the correlation coefficient is

$$\rho_{12} = -1, \tag{9.38}$$

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 not attributable to the design of the filter. There are *two* unknowns (x_1^1 and x_1^2) and *one* constraint:

$$z_1 = (x_1^1 + x_1^2). \tag{9.39}$$

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

$$C \stackrel{\text{def}}{=} \frac{x_1^2}{x_1^1} \tag{9.40}$$

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 these conditions are satisfied *only if*

$$\frac{\sigma_2^2(0)}{\sigma_1^2(0)} = C = \frac{x_1^2}{x_1^1} \geq 0, \tag{9.41}$$

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" of 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 9.9 (Unobservability Caused by Poor Choice of Sampling Rate) The problem in Example 9.8 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 parameters (constant states) with both constant and sinusoidal measurement sensitivity matrix components:

$$H(t) = [1 \quad \cos(\omega t)],$$

as plotted in Figure 9.5. 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 sensitivity matrix become equal and constant:

$$\begin{aligned} H_k &= [1 \quad \cos(\omega kT)] \\ &= [1 \quad \cos(2\pi k)] \\ &= [1 \quad 1], \end{aligned}$$

as shown in Figure 9.5. (This is the way many engineers discover "aliasing.") The states x^1 and x^2 are *unobservable* with this choice of sampling interval (see

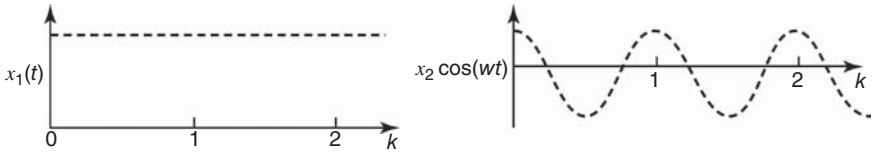


Figure 9.5 Aliased measurement components.

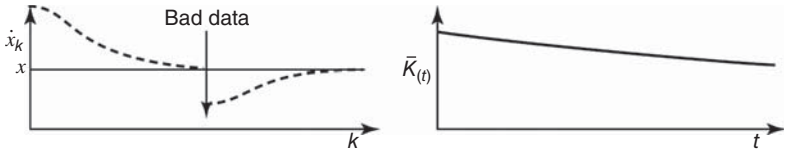


Figure 9.6 Asymptotic recovery from bad data.

Figure 9.5). 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 9.8 plus the more obvious remedy of changing the sampling interval T to obtain observability, for example,

$$T = \frac{\pi}{\omega} \tag{9.42}$$

is a better choice.

Causes of Unpredicted Nonconvergence Unpredictable nonconvergence may be caused by

1. bad data,
2. numerical problems, or
3. mismodeling.

Example 9.10 (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,

$$|\hat{x}(0) - x|^2 = |\tilde{x}|^2 \gg \text{trace} P_0. \tag{9.43}$$

- The measurement has an exogenous component (a mistake, not an error) that is excessively large, for example,

$$|v|^2 \gg \text{trace} R. \tag{9.44}$$

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 9.6.

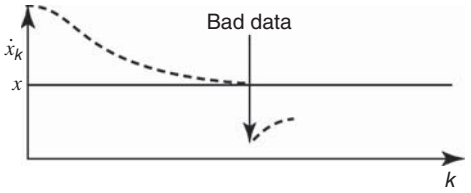


Figure 9.7 Failure to recover in short period.

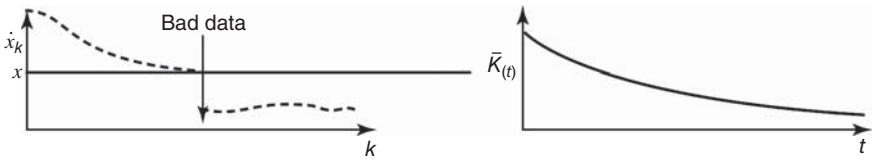


Figure 9.8 Failure to recover due to gain decay.

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 short to allow sufficient recovery (see Figure 9.7). The normal behavior of the gain matrix $\overline{K}$ may be too rapidly converging toward its steady-state value of $\overline{K} = 0$. (See Figure 9.8.)

9.2.4.3 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 it has spoiled the estimate (see Section 9.3).

The best remedy for this problem is to implement a "bad data detector" to reject the bad data before it contaminates the estimate. If this is to be done in real time, it is sometimes useful to *save* the bad data for offline 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.

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 9.11 (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 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 7) or eliminate the inverse altogether by diagonalizing R and processing the measurements sequentially (also shown in Chapter 7).

9.2.5 Effects Due to Mismodeling

9.2.5.1 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,
3. errors in the dynamic coefficients or state-transition matrix, and
4. overlooked nonlinearities.

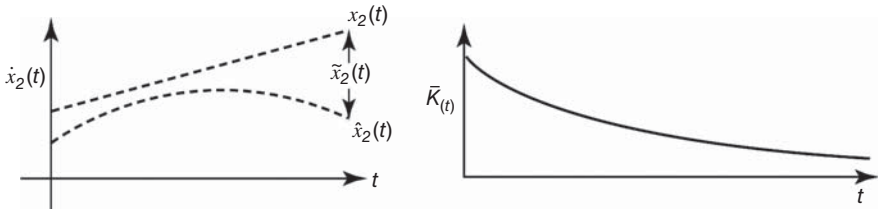


Figure 9.9 Divergence due to mismodeling.

Example 9.12 (Nonconvergence Caused by Unmodeled State Variables) Consider the following example:

Real-World (Creeping State)	Kalman Filter Model (Constant State)
$\dot{x}_1 = 0$	$\dot{x}_2 = 0$
$\dot{x}_2 = x_1$	$z = x_2 + v$
$z = x_2 + v$	
$\Rightarrow x_2(t) = x_2(0) + x_1(0)t$	$\Rightarrow x_2(t) = x_2(0)$

(9.45)

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 9.9.

Detecting Unmodeled State Dynamics by Fourier Analysis of the Filter Innovations It is difficult to diagnose unmodeled state variables unless all other causes for 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 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 would indicate the characteristic frequencies of unmodeled effects. These peaks can be at zero frequency, which would indicate 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 9.13 (Adding "Fictitious" Process Noise to the Kalman Filter Model) Continuing with the continuous-time problem of Example 7.10, 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 5, with parameters

$$F = 0, \quad H = 1, \quad G = 1,$$

transforming the general Riccati differential equation

$$\begin{aligned}\dot{P} &= FP + PF^{\mathrm{T}} - PH^{\mathrm{T}}R^{-1}HP + GQG^{\mathrm{T}} \\ &= \frac{-P^2}{R} + Q\end{aligned}$$

to a scalar equation with steady-state solution (to $\dot{P} = 0$)

$$P(\infty) = \sqrt{RQ}.$$

The steady-state Kalman gain

$$\overline{K}(\infty) = \frac{P(\infty)}{R} = \sqrt{\frac{Q}{R}}. \tag{9.46}$$

The equivalent steady-state model of the Kalman filter can now be formulated as follows:[2]

$$\dot{\hat{x}}_2 = F\hat{x}_2 + \overline{K}[z - H\hat{x}_2]. \tag{9.47}$$

Here, $F = 0, H = 1, \overline{K} = \overline{K}(\infty), \hat{x} = \hat{x}_2$, so that

$$\dot{\hat{x}}_2 + \overline{K}(\infty)\hat{x}_2 = \overline{K}(\infty)z. \tag{9.48}$$

The steady-state response of this estimator can be determined analytically by taking Laplace transforms:

$$[s + \overline{K}(\infty)]\hat{x}_2(s) = \overline{K}(\infty)z(s) \tag{9.49}$$

$$\Rightarrow \frac{\hat{x}_2(s)}{z(s)} = \frac{\overline{K}(\infty)}{s + \overline{K}(\infty)}. \tag{9.50}$$

Figure 9.10(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:

$$z(t) = x_2(t) = x_2(0) + x_1(0)t \quad \text{with } v = 0.$$

Taking the Laplace transform of the equation yields

$$z(s) = x_2(s) = \frac{x_2(0)}{s} + \frac{x_1(0)}{s^2}.$$

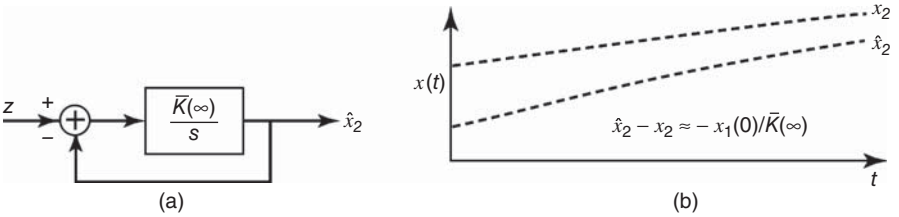


Figure 9.10 Type-1 servo (a) and estimates (b).

The error in $x_2(t)$ is

$$\tilde{x}_2(t) = \hat{x}_2(t) - x_2(t).$$

Taking the Laplace of the equation and substituting the value of $\hat{x}_2(s)$ from Equation 9.50 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 9.10(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 9.14 (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
$\dot{x}_1 = 0$	$\dot{x}_1 = w$
$\dot{x}_2 = x_1$	$\dot{x}_2 = x_1$
$z = x_2 + v$	$z = x_2 + v$

(9.51)

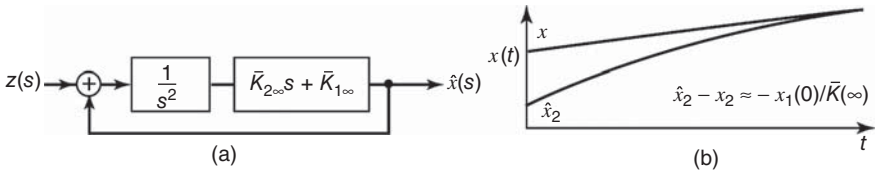


Figure 9.11 Type-2 servo (a) and servo estimates (b).

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 9.11(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,

$$F = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}, \quad G = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad H = [0 \quad 1], \quad Q = \text{cov}(w), \quad R = \text{cov}(v) \tag{9.52}$$

$$\dot{P} = FP + PF^{\mathrm{T}} + GQG^{\mathrm{T}} - PH^{\mathrm{T}}R^{-1}HP \text{ and } \overline{K} = PH^{\mathrm{T}}R^{-1} \tag{9.53}$$

become in the steady state

$$\left.\begin{aligned} \dot{P}_{11} &= Q - \frac{p_{12}^2}{R} = 0 \\ \dot{P}_{12} &= p_{11} - \frac{p_{12}p_{22}}{R} = 0 \\ \dot{P}_{22} &= 2p_{12} - \frac{p_{22}^2}{R} = 0 \end{aligned}\right\} \Rightarrow \begin{aligned} p_{12}(\infty) &= \sqrt{RQ}, \\ p_{22}(\infty) &= \sqrt{2}(R^3Q)^{1/4}, \\ p_{11}(\infty) &= \sqrt{2}(Q^3R)^{1/4}, \\ \overline{K}(\infty) &= \begin{bmatrix} \sqrt{\frac{Q}{R}} \\ \sqrt{2}\sqrt[4]{\frac{Q}{R}} \end{bmatrix} = \begin{bmatrix} \overline{K}_1(\infty) \\ \overline{K}_2(\infty) \end{bmatrix}, \end{aligned} \tag{9.54}$$

and these can be Laplace transformed to yield

$$\hat{x}(s) = [sI - F + \overline{K}(\infty)H]^{-1}\overline{K}(\infty)z(s). \tag{9.55}$$

In component form, this becomes

$$\hat{x}_2(s) = \frac{(\overline{K}_2 s + \overline{K}_1)/s^2}{1 + (\overline{K}_2 s + \overline{K}_1)/s^2} z(s). \tag{9.56}$$

The resulting steady-state following error to a ramp (in the noise-free case) is easily determined:

$$z(t) = x_2(t) = x_2(0) + x_1(0)t, \quad v = 0,$$

$$\tilde{x}_2(s) = \hat{x}_2(s) - x_2(s) = -\left(\frac{s^2}{s^2 + \overline{K}_2 s + \overline{K}_1}\right) x_2(s),$$

$$\tilde{x}_2(\infty) = \lim_{s\to 0} s\left(\frac{-s^2}{s^2 + \overline{K}_2 s + \overline{K}_1}\right)\left(\frac{x_2(0)}{s} + \frac{x_1(0)}{s^2}\right) = 0.$$

This type of behavior is shown in Figure 9.11(b).

Example 9.15 (Statistical Modeling Errors Due to Unmodeled Dynamic Distubrance Noise) With the models

$$\begin{array}{cc} \underline{\text{Real World}} & \underline{\text{Kalman Filter Model}} \\ \dot{x}_1 = w & \dot{x}_1 = 0 \\ \dot{x}_2 = x_1 & \dot{x}_2 = x_1 \\ z = x_2 + v & z = x_2 + v \end{array} \tag{9.57}$$

the Kalman filter equations yield the following relationships:

$$\left.\begin{array}{l} \dot{p}_{11} = \frac{-1}{R} p_{12}^2 \\ \dot{p}_{12} = p_{11} - \dfrac{p_{12}p_{22}}{R} \\ \dot{p}_{22} = 2p_{12} - \frac{p_{22}^2}{R} \end{array}\right\} \quad \Rightarrow \quad \text{in the steady state:} \quad \begin{array}{l} p_{12} = 0, \quad \overline{K} = PH^{\mathrm{T}}R^{-1} = 0, \\ p_{11} = 0, \quad \hat{x}_1 = \text{const}, \\ p_{22} = 0, \quad \hat{x}_2 = \text{ramp}. \end{array} \tag{9.58}$$

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 9.12. 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 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 particular example, $\sigma_{x_1}^2(t) = \sigma_{x_1}^2(0) + \sigma_w^2 t$. 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" dynamic disturbance noise.

Example 9.16 (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:

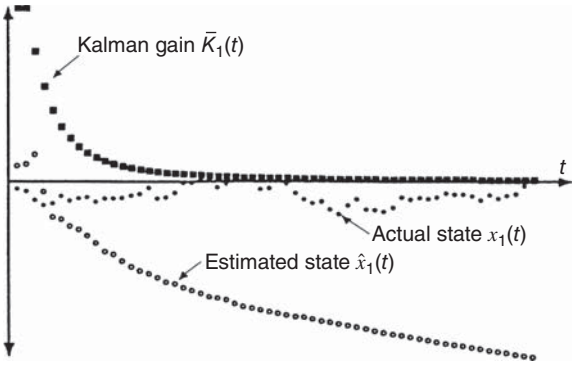


Figure 9.12 Diverging estimation error due to unmodeled process noise.

$$\begin{array}{cc}
\underline{\text{Real World}} & \underline{\text{Kalman Filter Model}} \\
\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix} = 0 & \begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix} = 0 \\
z = [\sin \Omega t | \cos \Omega t] \begin{bmatrix}x_1\\ x_2\end{bmatrix} + v & z = [\sin \omega t | \cos \omega t] \begin{bmatrix}x_1\\ x_2\end{bmatrix} + v \\
v = \text{white noise} & v = \text{white noise} \\
\text{No a priori information exists} & \text{No a priori information exists}
\end{array} \tag{9.59}$$

In this case, the optimum filter is the "least-squares" estimator acting over the measurement interval $(0, T)$. Since this is a continuous case,

$$J = \int_0^{\mathrm{T}} (z - H\hat{x})^{\mathrm{T}}(z - H\hat{x}) \quad dt \tag{9.60}$$

is the performance index being minimized. Its gradient

$$\frac{\partial J}{\partial \hat{x}} = 0 \quad \Rightarrow \quad 0 = 2\int_0^{\mathrm{T}} H^{\mathrm{T}} z \quad dt - 2\int_0^{\mathrm{T}} (H^{\mathrm{T}}H)\hat{x} \quad dt, \tag{9.61}$$

where $\hat{x}$ is unknown but constant. Therefore,

$$\hat{x} = \left[\int_0^{\mathrm{T}} H^{\mathrm{T}}H \quad dt\right]^{-1} \left[\int_0^{\mathrm{T}} H^{\mathrm{T}} z \quad dt\right], \tag{9.62}$$

where

$$H = [\sin \omega t | \cos \omega t]; \qquad z = [\sin \Omega t | \cos \Omega t] \begin{bmatrix}x_1\\ x_2\end{bmatrix} + v \tag{9.63}$$

and

$$\omega = 2\pi f = \frac{2\pi}{p}, \tag{9.64}$$

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

$$\hat{x} = \begin{bmatrix} \frac{Np}{2} & 0 \\ 0 & \frac{Np}{2} \end{bmatrix}^{-1} \begin{bmatrix} \int_0^{Np} \sin(\omega t) z(t) & dt \\ \int_0^{Np} \cos(\omega t) z(t) & dt \end{bmatrix} = \begin{bmatrix} \frac{2}{Np} \int_0^{Np} z(t) \sin \omega t & dt \\ \frac{2}{Np} \int_0^{Np} z(t) \cos \omega t & dt \end{bmatrix}. \quad (9.65)$$

Concentrating on the first component $\hat{x}_1$, one can obtain its solution as

$$\begin{aligned} \hat{x}_1 = & \left\{ \frac{2}{Np} \left[\frac{\sin(\omega - \Omega)t}{2(\omega - \Omega)} - \frac{\sin(\omega + \Omega)t}{2(\omega + \Omega)} \right] \Big|_{t=0}^{t=Np} \right\} x_1 \\ & + \left\{ \frac{2}{Np} \left[\frac{-\cos(\omega - \Omega)t}{2(\omega - \Omega)} - \frac{\cos(\omega + \Omega)t}{2(\omega + \Omega)} \right] \Big|_{t=0}^{t=Np} \right\} x_2 \\ & + \frac{2}{Np} \int_0^{Np} v(t) \sin \omega t \quad dt. \end{aligned}$$

By setting $v = 0$ (ignoring the estimation error due to measurement noise), one obtains the result

$$\begin{aligned} \hat{x}_1 = & \frac{2}{Np} \left[\frac{\sin(\omega - \Omega)Np}{2(\omega - \Omega)} - \frac{\sin(\omega + \Omega)Np}{2(\omega + \Omega)} \right] x_1 \\ & + \frac{2}{Np} \left[\frac{1 - \cos(\omega - \Omega)Np}{2(\omega - \Omega)} + \frac{1 - \cos(\omega + \Omega)t}{2(\omega + \Omega)} \right] x_2. \end{aligned}$$

For the case that $\omega \to \Omega$,

$$\left.\begin{aligned} \frac{\sin(\omega - \Omega)Np}{2(\omega - \Omega)} &= \frac{Np}{2} \frac{\sin x}{x}\Big|_{x \to 0} = \frac{Np}{2}, \\ \frac{\sin(\omega + \Omega)Np}{2(\omega + \Omega)} &= \frac{\sin[(4\pi/p)Np]}{2\ \Omega} = 0, \\ \frac{1 - \cos(\omega - \Omega)Np}{2(\omega - \Omega)} &= \frac{1 - \cos x}{x}\Big|_{x \to 0} = 0, \\ \frac{1 - \cos(\omega + \Omega)Np}{2(\omega + \Omega)} &= \frac{1 - \cos[(4\pi/p)Np]}{2\ \Omega} = 0, \end{aligned}\right\} \quad (9.66)$$

and $\hat{x}_1 = x_1$. In any other case, $\hat{x}_1$ would be a biased estimate of the form

$$\hat{x}_1 = \Upsilon_1 x_1 + \Upsilon_2 x_2, \qquad \text{where} \quad \Upsilon_1 \neq 1, \quad \Upsilon_2 \neq 0. \quad (9.67)$$

Similar behavior occurs with $\hat{x}_2$.

Wrong parameters in the system and/or output matrices may not cause the filter covariance matrix or state vector to look unusual. However, the innovations vector $[z - H\hat{x}]$ will generally show detectable effects of nonconvergence.

This can only be cured 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 9.17 (Parameter Estimation) This reformulation provides an example of a nonlinear estimation problem:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = 0, \qquad x_3 = \Omega. \tag{9.68}$$

Here, something is known about Ω, but it is not known precisely enough. One must choose

$$\hat{x}_3(0) = \text{"best guess" value of } \Omega,$$
$$P_{33}(0) = \sigma^2_{\tilde{x}_3}(0) = \text{a measure of uncertainty in } \tilde{x}_3(0).$$

Nonlinearities in the real-world system also cause nonconvergence or even divergence of the Kalman estimates.

9.2.6 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.

9.2.6.1 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 9.18 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 of the test methods, because it yields the actual eigenvalues (characteristic values) and their associated eigenvectors (characteristic vectors). The characteristic vectors tell 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:

$$P = TDT^{\mathrm{T}} \text{ (symmetric eigenvalue–eigenvector decomposition)}, \tag{9.69}$$

$$D = \mathrm{diag}_i\{d_i\}, \tag{9.70}$$

$$d_1 \geq d_2 \geq d_3 \geq \cdots \geq d_n. \tag{9.71}$$

If

$$d_n < 0, \tag{9.72}$$

then replace P with

$$P^* = TD^*T^{\mathrm{T}}, \tag{9.73}$$

$$D^* = \mathrm{diag}_i\{d_i^*\}, \tag{9.74}$$

$$d_i^* = \begin{cases} d_i & \text{if} \quad d_i \geq 0, \\ 0 & \text{if} \quad d_i < 0. \end{cases} \tag{9.75}$$

9.3 PREFILTERING AND DATA REJECTION METHODS

9.3.1 Prefiltering

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 9.13 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 that 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 [2].
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.

9.3.1.1 Continuous (Analog) Linear Filters

Example 9.19 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 9.14. An example of such a prefilter is a digital voltmeter (DVM). A DVM is essentially a time-gated averager with sampler and quantizer.

Thus the input signal is continuous and the output discrete. A functional representation is given in Figures 9.15–9.17, where

$$\Delta T = \text{sampling interval},$$

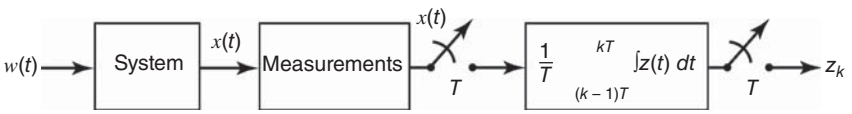


Figure 9.13 Integrate-and-hold prefilter.

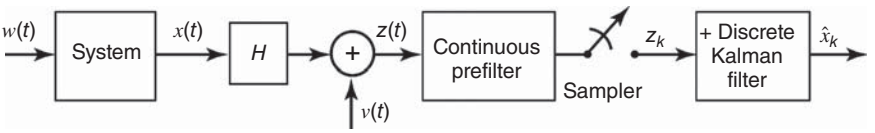


Figure 9.14 Continuous linear prefiltering and sampling.

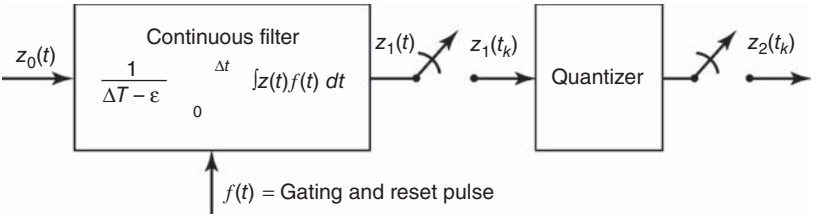


Figure 9.15 Block diagram of DVM.

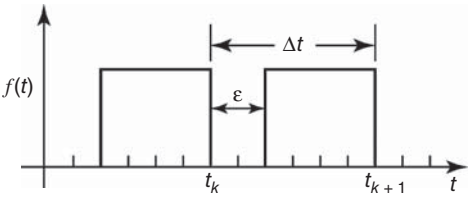


Figure 9.16 DVM gating waveform.

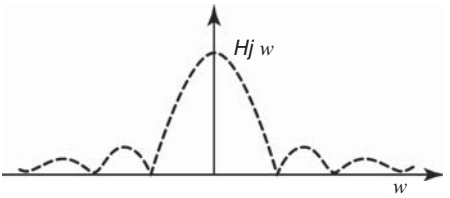


Figure 9.17 DVM frequency response.

ε = dead time for rezeroing the integrator,

$\Delta T - \varepsilon$ = averaging time,

and the output

$$z^1(t_i) = \frac{1}{\Delta T - \varepsilon} \int_{t_i - \Delta T + \varepsilon}^{t_i} z(t) \quad dt. \tag{9.76}$$

It can be shown that the frequency response of the DVM is

$$|H(j\omega| = \left| \frac{\sin \omega[(\Delta T - \varepsilon)/2]}{\omega[(\Delta T - \varepsilon)/2]} \right| \quad \text{and} \quad \theta(j\omega) = -\omega \left(\frac{\Delta T - \varepsilon}{2} \right). \tag{9.77}$$

With white-noise continuous input, the output is a white-noise sequence (since the averaging intervals are nonoverlapping). With an exponentially correlated random

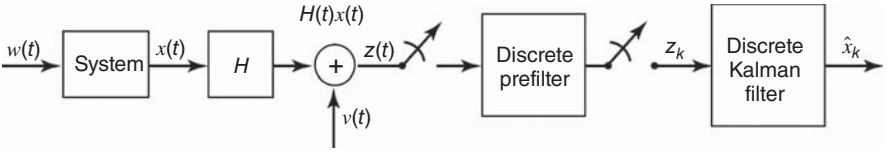


Figure 9.18 Discrete linear prefilter.

continuous input with correlation time τ_c and autocovariance

$$\psi_z(\tau) = \sigma_z^2 e^{-(|\tau|/\tau_c)}, \tag{9.78}$$

the variance and autocorrelation function of the output random sequence can be shown to be

$$\left.\begin{aligned} \sigma_{z^1}^2 &= \psi_{z^1 z^1}(0) = f(u)\sigma_z^2, \\ \psi(j-i) &= g(u)\sigma_z^2 e^{-(j-i)\frac{\Delta T}{\tau_c}}, \end{aligned}\right\} \quad \begin{aligned} u &= \tfrac{\Delta T - \varepsilon}{T_c}, \\ f(u) &= \tfrac{2}{u^2}(e^{-u} + u - 1) \\ g(u) &= \tfrac{e^u + e^{-u} - 2}{u^2}. \end{aligned} \tag{9.79}$$

9.3.1.2 Discrete Linear Filters These can be used effectively for the attenuation of the effects of some of the state variables to the point that 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 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 9.18).

Example 9.20 For the simple digital averager shown in Figure 9.19, let

$$z^1(t_i^1) \equiv \frac{1}{N} \sum_{j=i-N+1}^{i} z(t_j), \quad t_i^1 = iT^1, \tag{9.80}$$

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

$$|H(j\omega)| = \frac{|\sin(N\omega T/2)|}{N|\sin(\omega T/2)|}, \theta(j\omega) = -\left(\frac{N-1}{2}\right)\omega T. \tag{9.81}$$

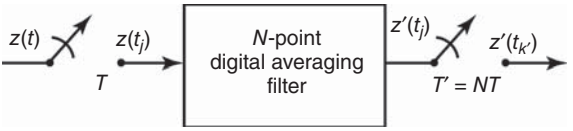


Figure 9.19 Discrete linear averager.

9.3.2 Data Rejection

If adequate knowledge of the innovations vector $[z - H\hat{x}]$ exists, nonlinear "data rejection filters" can be implemented. Statistical approaches were discussed in Section 9.2.1. Some simple examples are cited below.

Example 9.21 (Data Rejection Filters) For excess amplitude,

$$\text{If } |(z - H\hat{x})_i| > A_{\max}, \text{ then reject data.} \tag{9.82}$$

For excess rate (or change),

$$\text{If } |(z - H\hat{x})_{i+1} - (z - H\hat{x})_i| > \delta A_{\max}, \text{ then reject data.} \tag{9.83}$$

Many other ingenious techniques have been used, but they usually depend on the specifics of the problem.

9.4 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.[4]

If the actual measurement processing in the Kalman filter state equations is neglected, then the resulting equations characterize the stability of the filter itself. In the continuous case, these equations are

$$\dot{\hat{x}}(t) = [F(t) - \overline{K}(t)H(t)]\hat{x}(t), \tag{9.84}$$

and in the discrete case,

$$\begin{aligned}\hat{x}_{k(+)} &= \Phi_{k-1}\hat{x}_{k-1(+)} - \overline{K}_k H_k \Phi_{k-1}\hat{x}_{k-1(+)} \\ &= [I - \overline{K}_k H_k]\Phi_{k-1}\hat{x}_{k-1(+)}.\end{aligned} \tag{9.85}$$

[4]See, for example, Gelb et al. [3, pp. 22, 31, 36, 53, 72] or Maybeck [4, p. 278].

The solution of the filter Equation 9.84 or 9.85 is uniformly asymptotically stable, which means bounded input-bounded output (BIBO) stability. Mathematically, it implies that

$$\lim_{t\to\infty} \|\hat{x}(t)\| = 0 \tag{9.86}$$

or

$$\lim_{k\to\infty} \|\hat{x}_k(+)\| = 0, \tag{9.87}$$

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 5 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 9.2).

9.5 SUBOPTIMAL AND REDUCED-ORDER FILTERS

9.5.1 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*.

9.5.1.1 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 5–7 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.

9.5.1.2 Suboptimal Filtering Techniques 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.

9.5.1.3 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 $\overline{\boldsymbol{K}}(t)$ *or* $\overline{\boldsymbol{K}}_{\boldsymbol{k}}$ Consider the system

$$\begin{aligned} \dot{x} &= Fx + Gw, \qquad Q = \text{cov}(w), \\ z &= Hx + v, \quad R = \text{cov}(v), \end{aligned} \tag{9.88}$$

with state-transition matrix Φ.

The state estimation portion of the optimal linear estimation algorithm is then, for the continuous case,

$$\dot{\hat{x}} = F\hat{x} + \overline{K}(t)[z - H\hat{x}] \text{ with } \hat{x}(0) = \hat{x}_0, \tag{9.89}$$

and for the discrete case,

$$\hat{x}_{k(+)} = \Phi_{k-1}\hat{x}_{k-1(+)} + \overline{K}_k[z_k - H_k(\Phi_{k-1}\hat{x}_{k-1(+)})] \tag{9.90}$$

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 9.20.

The Wiener filter results from the approximation

$$\overline{K}(t) \approx \overline{K}(\infty). \tag{9.91}$$

If, in addition, the matrices F and H are time invariant, the matrix of transfer functions characterizing the Wiener filter can easily be computed:

$$\dot{\hat{x}} = F\hat{x} + \overline{K}(\infty)[z - H\hat{x}] \tag{9.92}$$

$$\Rightarrow \frac{\hat{x}(s)}{z(s)} = [sI - F + \overline{K}(\infty)H]^{-1}. \tag{9.93}$$

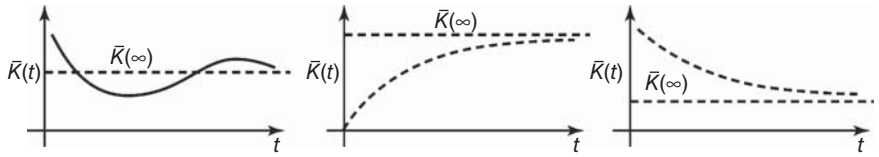


Figure 9.20 Settling of Kalman gains.

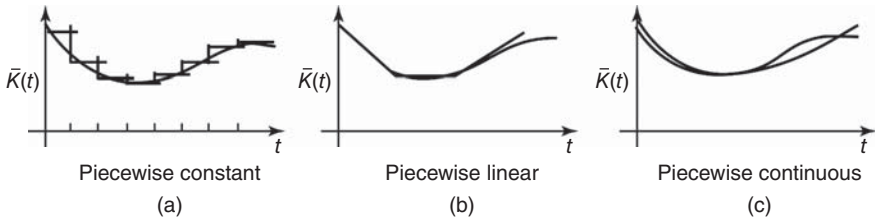


Figure 9.21 Approximating time-varying Kalman gains. (a) Piecewise constant, (b) Piecewise linear, and (c) Piecewise continuous.

The corresponding steady-state sinusoidal frequency response matrix is

$$\frac{|\hat{x}(j\omega)|}{|z(j\omega)|} = |[(j\omega)I - F + \overline{K}(\infty)H]^{-1}|. \tag{9.94}$$

Among the advantages of the Wiener filter are that—its structure being identical to 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 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 $\overline{K}_{\text{OP}}(t)$ are often approximated by simple functions.

For example, one can use piecewise constant approximation, $\overline{K}_{\text{pwc}}$, a piecewise linear approximation, $\overline{K}_{\text{pwl}}$, or a curve fit using smooth functions $\overline{K}_{\text{CF}}$, as shown in Figure 9.21:

$$\overline{K}_{\text{CF}}(t) = C_1 e^{-a_1 t} + C_2(1 - e^{-a_2 t}). \tag{9.95}$$

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.

9.5.1.4 Modifying Filter Models 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 with frequency-domain approximations. Ignoring some states reduces model complexity and provides a suboptimal design. Often, however, little performance degradation occurs.

Example 9.22 In this example, one combines two nonobservable states with identical propagation into z.

$$\begin{aligned} \text{From:} \quad & \dot{x}_1 = -ax_1, \\ & \dot{x}_2 = -ax_2, \\ & z = [23]\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + v, \end{aligned} \qquad \begin{aligned} \text{To:} \quad & \dot{x}^1 = -ax^1, \\ & z = x^1 + v, \end{aligned} \tag{9.96}$$

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 9.23 Continuing where Example 9.13 left off, one can combine two states if they are "close functions" over the entire measurement interval:

$$\begin{aligned} \text{From:} \quad & \dot{x}_1 = 0, \\ & \dot{x}_2 = 0, \\ & z = [t|\sin t]\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + v, \end{aligned} \qquad \begin{aligned} \text{To:} \quad & \dot{x}^1 = 0, \\ & z = tx^1 + v, \end{aligned} \tag{9.97}$$

where the a priori information on x^1 must be formulated and where z is available on the interval $(0, \pi/20.)$

Example 9.24 (Ignoring Small Effects)

$$\begin{aligned} \text{From:} \quad & \dot{x}_1 = -ax_1, \\ & \dot{x}_2 = -bx_2, \\ & z = x_1 + x_2 + v \text{ on } (0, T), \end{aligned} \qquad \begin{aligned} \text{To:} \quad & \dot{x}_2 = -bx_2, \\ & z = x_2 + v, \end{aligned} \tag{9.98}$$

with

$$E(x_1^2) = 0.1, \quad E(x_2^2) = 10.0, \qquad \text{desired} \quad P_{22}(T) = 1.0. \tag{9.99}$$

Example 9.25 (Relying on Time Orthogonality of States)

$$
\begin{aligned}
\text{From:}\quad & \dot{x}_1 = 0, \\
& \dot{x}_2 = 0, \\
& z = [1 \mid \sin t] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + v, \\
& z \text{ available on } (0, T) \text{ with } T \text{ large} \\
& P_{22}(0) = \text{large}.
\end{aligned}
\qquad
\begin{aligned}
\text{To:}\quad & \dot{x}_2 = 0, \\
& z = (\sin t) x_2 + v,
\end{aligned}
\tag{9.100}
$$

9.5.1.5 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 9.22.

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 shape of signals). The filter may be modified as shown in Figure 9.23 to compensate for such effects. Hopefully, the net result is a smaller filter than before.

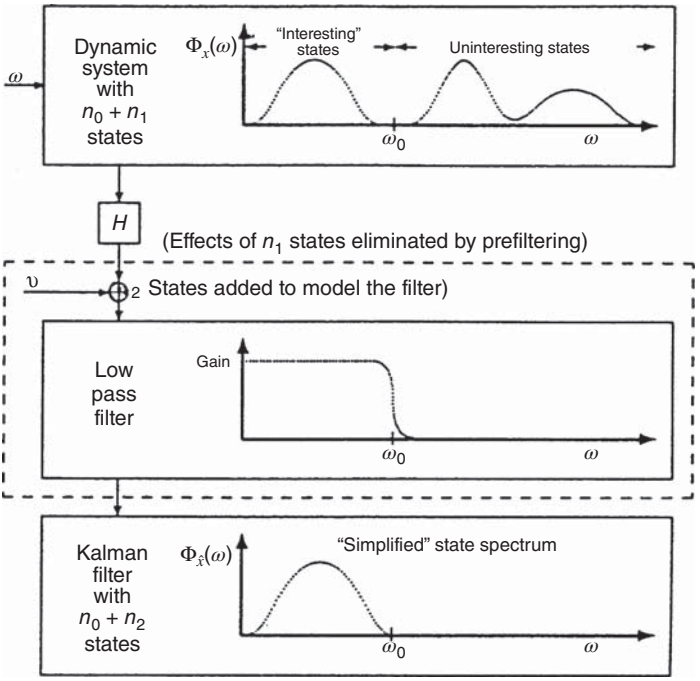


Figure 9.22 Low pass prefiltering for model simplification.

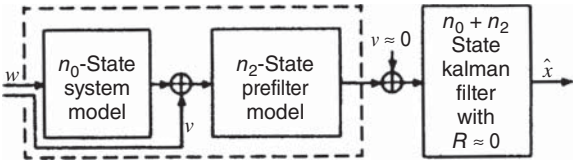


Figure 9.23 Modified system model for low pass prefiltering.

Example 9.26 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:

$$\begin{aligned} \dot{x}_1 &= -\alpha_1 x_1 + w_1, \quad & \dot{x}_2 &= -\alpha_2 x_2 + w_2, \\ z_1 &= x_1 + v_1, & z_2 &= x_2 + v_2. \end{aligned} \tag{9.101}$$

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.

9.5.1.6 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 power spectral densities can be approximated as shown in Figure 9.24.

A filter designed for the approximate spectrum may well use fewer states.

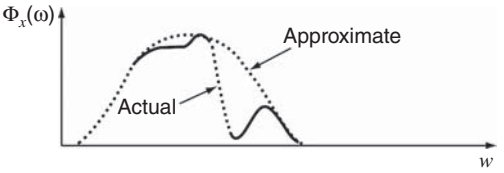


Figure 9.24 Frequency-domain approximation.

Example 9.27 Sometimes the general structure of the random process model is known, but the parameters are not known precisely:

$$\dot{x} = -\alpha x + w \text{ and } \sigma_w \text{ are "uncertain."} \tag{9.102}$$

Replacing this model by a random walk

$$\dot{x} = w \tag{9.103}$$

in conjunction with "sensitivity studies" will often allow a judicious choice for σ_w with small sensitivity to α uncertainties and small degradation in filter performance.

Example 9.28 (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 9.25.

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.

9.5.2 Dual-State Evaluation of Suboptimal Filters

9.5.2.1 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.

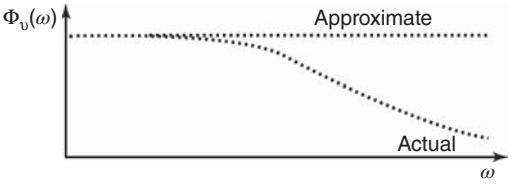


Figure 9.25 White-noise approximation of broadband noise.

2. The *filter model*, which is a reduced-order version of the system model, it 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 just 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*.

9.5.2.2 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 9.31.[5] 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.

In dual-state analysis for evaluating suboptimal filters, let the following definitions apply:

Actual System	Filter Model	
$\dot{x}^S = F_S x^S + G_S w^S$	$\dot{x}^F = F_F x^F + G_F w^F$	(9.104)
$z^S = H_S x^S + v^S$	$z^F = H_F x^F + v^F$	
$z^F = z^S,$	$H_F = H_S, v^F = v^S,$	(9.105)

[5]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.

$$\left.\begin{array}{l}\eta_k^S \equiv \int_{t_{k-1}}^{t_k} \Phi_S(t_k - \tau) G_S(\tau) w^S(\tau)\, \mathrm{d}\tau, \\ \hat{x}_{k(+)}^F = \Phi_F \hat{x}_{k-1(+)}^F + \overline{K}_k [z_k^F - H_F \Phi_F \hat{x}_{k-1(+)}^F], \\ \Phi_S \equiv \text{System state-transition matrix}, \\ \Phi_F \equiv \text{State-transition matrix of filter model}, \\ Q_S \equiv \text{cov}(w^S), \\ Q_F \equiv \text{cov}(w^F), \end{array}\right\} \tag{9.106}$$

Let $\Gamma_k \equiv (I - \overline{K}_k H_F)$, and let

$$A \equiv \begin{bmatrix} \Phi_F & \Phi_F - \Phi_S \\ 0 & \Phi_S \end{bmatrix}, \tag{9.107}$$

$$B \equiv \begin{bmatrix} \Gamma_k & 0 \\ 0 & I \end{bmatrix}. \tag{9.108}$$

Let the "prediction" estimation error be

$$\tilde{x}_k^S(+) \equiv \hat{x}_k^F(+) - x_k^S \tag{9.109}$$

and let the "filtered" estimation error be

$$\tilde{x}_{k-1}^S(+) \equiv \hat{x}_{k-1}^F(+) - x_{k-1}^S. \tag{9.110}$$

Then the prediction error equation is

$$\begin{bmatrix} \tilde{x}_{k(-)}^S \\ x_k^S \end{bmatrix} = A \begin{bmatrix} \tilde{x}_{k-1(+)}^S \\ x_{k-1}^S \end{bmatrix} + \begin{bmatrix} -\eta_{k-1}^S \\ \eta_{k-1}^S \end{bmatrix} \tag{9.111}$$

and the filtered error equation is

$$\begin{bmatrix} \tilde{x}_{k-1(+)}^S \\ x_{k-1}^S \end{bmatrix} = B \begin{bmatrix} \tilde{x}_{k-1(-)}^S \\ x_{k-1}^S \end{bmatrix} + \begin{bmatrix} \overline{K}_{k-1} v_{k-1}^S \\ 0 \end{bmatrix}. \tag{9.112}$$

Taking the expected value E and the covariance of Equations 9.111 and 9.112 yields the following recursive relationships:

$$\left.\begin{array}{l} E_{\text{cov}} = A \cdot P_{\text{cov}} \cdot A^T + \text{cov}L \\ P_{\text{cov}} = B \cdot E_{\text{cov}} \cdot B^T + \text{cov}P \end{array}\right\} \text{(covariance propagation)}, \tag{9.113}$$

$$\left.\begin{array}{l} E_{\text{DX}} = A \cdot P_{\text{DX}} \\ P_{\text{DX}} = B \cdot E_{\text{DX}} \end{array}\right\} \text{(bias propagation)}, \tag{9.114}$$

where the newly introduced symbols are defined as follows:

$$
\begin{aligned}
\mathrm{E}_{\mathrm{cov}} &\equiv \mathrm{cov}\begin{bmatrix}\tilde{x}^S_{k(-)}\\ x^S_k\end{bmatrix} \text{ (predicted dual-state vector)},\\
\mathrm{cov}L &\equiv \mathrm{cov}\begin{bmatrix}-\eta^S_{k-1}\\ \eta^S_{k-1}\end{bmatrix},\\
P_{\mathrm{cov}} &\equiv \mathrm{cov}\begin{bmatrix}\tilde{x}^S_{k-1(+)}\\ x^S_{k-1}\end{bmatrix} \text{ (filtered covariance)},\\
\mathrm{cov}P &\equiv \mathrm{cov}\begin{bmatrix}\overline{K}_{k-1}v^S_{k-1}\\ 0\end{bmatrix},\\
&= \begin{bmatrix}\overline{K}_{k-1}\mathrm{cov}(v^S_{k-1})\overline{K}^{\mathrm{T}}_{k-1} & 0\\ 0 & 0\end{bmatrix},\\
P_{\mathrm{DX}} &= \mathrm{E}\begin{bmatrix}\tilde{x}^S_{k-1(+)}\\ x^S_{k-1}\end{bmatrix} \text{ (expected value of filtered dual-state vector)},\\
\mathrm{E}_{\mathrm{DX}} &= \mathrm{E}\begin{bmatrix}\tilde{x}^S_{k(-)}\\ x^S_k\end{bmatrix} \text{ (expected value of predicted dual-state vector)}.
\end{aligned}
\tag{9.115}
$$

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

$$E_{\mathrm{DX}} = A \cdot P_{\mathrm{DX}}, P_{\mathrm{DX}} = B \cdot E_{\mathrm{DX}}$$

become

$$\mathrm{E}\begin{bmatrix}\tilde{x}^S_{k(-)}\\ x^S_k\end{bmatrix} = A\mathrm{E}\begin{bmatrix}\tilde{x}^S_{k-1(+)}\\ x^S_{k-1}\end{bmatrix} \tag{9.116}$$

and

$$\mathrm{E}\begin{bmatrix}\tilde{x}^S_{k-1(+)}\\ x^S_{k-1}\end{bmatrix} = B\mathrm{E}\begin{bmatrix}\tilde{x}^S_{k-1(-)}\\ x^S_{k-1}\end{bmatrix}. \tag{9.117}$$

Clearly, if $\mathrm{E}[\tilde{x}^S_k(+)] \neq 0$, then the estimate becomes biased. If

$$\Phi_F \neq \Phi_S \tag{9.118}$$

and

$$\mathrm{E}(x^S) \neq 0, \tag{9.119}$$

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

$$E_{\text{cov}} = A(P_{\text{cov}})A^{\text{T}} + \text{cov}L$$
$$P_{\text{cov}} = B(E_{\text{cov}})B^{\text{T}} + \text{cov}P$$

show that $\text{cov}[\tilde{x}_k^S(-)]$ depends on $\text{cov}(x_k^S)$. (See Problem 7.5.)

If

$$\Phi_F \neq \Phi_S \tag{9.120}$$

and

$$\text{cov}(x^S) \neq 0, \tag{9.121}$$

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.

9.6 SCHMIDT–KALMAN FILTERING

9.6.1 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 Reference 5, 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 could be reordered in the state vector such that the nuisance variables appear last. The resulting dynamic equation then has the form

$$\frac{d}{dt}\mathbf{x}(t) = \begin{bmatrix} F_{\varepsilon}(t) & 0 \\ 0 & F_{\nu}(t) \end{bmatrix} \mathbf{x}(t) + \mathbf{w}(t) \tag{9.122}$$

such that F_{ν} 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*.

9.6.2 Derivation

9.6.2.1 Partitioning the Model Let[6]

$n = n_{\varepsilon} + n_{\nu}$ be the total number of state variables,

n_{ε} be the number of essential variables, whose values are of interest for the application, and

n_{ν} 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:

$$\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{n_{\varepsilon}} \\ x_{n_{\varepsilon+1}} \\ x_{n_{\varepsilon+2}} \\ x_{n_{\varepsilon+3}} \\ \vdots \\ x_{n_{\varepsilon}+n_{\nu}} \end{bmatrix} \begin{matrix} \left.\begin{matrix} \\ \\ \\ \\ \\ \end{matrix}\right\} \text{essential variables} \\ \left.\begin{matrix} \\ \\ \\ \\ \\ \end{matrix}\right\} \text{nuisance variables} \end{matrix} \tag{9.123}$$

[6]This derivation follows that in Reference 6.

$$= \left[\frac{\mathbf{x}_\varepsilon}{\mathbf{x}_\nu}\right], \tag{9.124}$$

where the state vector has been partitioned into a subvector $\mathbf{x}_\varepsilon$ of the essential state variables and a subvector $\mathbf{x}_\nu$ of nuisance variables.

9.6.2.2 Partitioning Dynamic Models We know that these two state variable types are not linked dynamically, so that the system dynamic model has the form

$$\frac{d}{dt}\left[\frac{\mathbf{x}_\varepsilon(t)}{\mathbf{x}_\nu(t)}\right] = \left[\frac{F_\varepsilon(t) \mid 0}{0 \mid F_\nu(t)}\right]\left[\frac{\mathbf{x}_\varepsilon(t)}{\mathbf{x}_\nu(t)}\right] + \left[\frac{\mathrm{w}_\varepsilon(t)}{\mathrm{w}_\nu(t)}\right] \tag{9.125}$$

in continuous time, where the process noise vectors $\mathbf{w}_\varepsilon$ and $\mathbf{w}_\nu$ are uncorrelated. That is, the covariance matrix of process noise

$$Q = \left[\frac{Q_{\varepsilon\varepsilon} \mid 0}{0 \mid Q_{\nu\nu}}\right] \tag{9.126}$$

for the continuous-time model (as well as for the discrete-time model). That is, the cross-covariance block $Q_{\varepsilon\nu} = 0$.

9.6.2.3 Partitioned Covariance Matrix The covariance matrix of estimation uncertainty (the dependent variable of the Riccati equation) can also be partitioned as

$$P = \left[\frac{P_{\varepsilon\varepsilon} \mid P_{\varepsilon\nu}}{P_{\nu\varepsilon} \mid P_{\nu\nu}}\right], \tag{9.127}$$

where

the block $P_{\varepsilon\varepsilon}$ is dimensioned $n_\varepsilon \times n_\varepsilon$,
the block $P_{\varepsilon\nu}$ is dimensioned $n_\varepsilon \times n_\nu$,
the block $P_{\nu\varepsilon}$ is dimensioned $n_\nu \times n_\varepsilon$, and
the block $P_{\nu\nu}$ is dimensioned $n_\nu \times n_\nu$.

9.6.2.4 Temporal Covariance Update The corresponding state-transition matrix for the discrete-time model will then be of the form

$$\Phi_\kappa = \left[\frac{\Phi_{\varepsilon\kappa} \mid 0}{0 \mid \Phi_{\nu\kappa}}\right], \tag{9.128}$$

$$\Phi_{\varepsilon k} = \exp\left(\int_{t_{k-1}}^{t_k} F_\varepsilon(t)\, dt\right), \tag{9.129}$$

$$\Phi_{\nu k} = \exp\left(\int_{t_{k-1}}^{t_k} F_\nu(t)\, dt\right), \tag{9.130}$$

and the temporal update of P will have the partitioned form

$$\left[\begin{array}{c|c} P_{\varepsilon\varepsilon k+1\,-} & P_{\varepsilon\nu k+1\,-} \\ \hline P_{\nu\varepsilon k+1\,-} & P_{\nu\nu k+1\,-} \end{array}\right] = \left[\begin{array}{c|c} \Phi_{\varepsilon k} & 0 \\ \hline 0 & \Phi_{\nu\,k} \end{array}\right] \left[\begin{array}{c|c} P_{\varepsilon\varepsilon k+} & P_{\varepsilon\nu k+} \\ \hline P_{\nu\varepsilon k+} & P_{\nu\nu k+} \end{array}\right] \left[\begin{array}{c|c} \Phi_{\varepsilon k}^{\mathrm{T}} & 0 \\ \hline 0 & \Phi_{\nu\,k}{}^{\mathrm{T}} \end{array}\right] + \left[\begin{array}{c|c} Q_{\varepsilon\varepsilon} & 0 \\ \hline 0 & Q_{\nu\nu} \end{array}\right], \tag{9.131}$$

or, in terms of the individual blocks,

$$P_{\epsilon\epsilon k+1-} = \Phi_{\epsilon k} P_{\epsilon\epsilon k+} \Phi_{\epsilon k}^{\mathrm{T}} + Q_{\epsilon\epsilon}, \tag{9.132}$$

$$P_{\epsilon\upsilon k+1-} = \Phi_{\epsilon k} P_{\epsilon\upsilon k+} \Phi_{\upsilon k}^{\mathrm{T}}, \tag{9.133}$$

$$P_{\upsilon\epsilon k+1-} = \Phi_{\upsilon k} P_{\upsilon\epsilon k+} \Phi_{\epsilon k}^{\mathrm{T}}, \tag{9.134}$$

$$P_{\upsilon\upsilon k+1-} = \Phi_{\upsilon k} P_{\upsilon\upsilon k+} \Phi_{\upsilon k}^{\mathrm{T}} + Q_{\upsilon\upsilon}. \tag{9.135}$$

9.6.2.5 Partitioned Measurement Sensitivity Matrix With this partitioning of the state vector, the measurement model will have the form

$$z = [H_\varepsilon \quad | \quad H_\upsilon] \left[\frac{\mathbf{x}_\varepsilon(t)}{\mathbf{x}_\upsilon(t)}\right] + \upsilon \tag{9.136}$$

$$= \underbrace{H_\epsilon \mathbf{x}_\varepsilon}_{\substack{\text{Essential}\\ \text{state}\\ \text{dependence}}} + \underbrace{H_\upsilon \mathbf{x}_\upsilon}_{\substack{\text{Correlated}\\ \text{noise}}} + \underbrace{\upsilon}_{\substack{\text{Uncorrelated}\\ \text{noise}}}. \tag{9.137}$$

9.6.3 Schmidt–Kalman Gain

9.6.3.1 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

$$\begin{aligned} \overline{K} &= \left[\frac{K_\epsilon}{K_\nu}\right] \\ &= \left[\begin{array}{c|c} P_{\varepsilon\epsilon} & P_{\epsilon\nu} \\ \hline P_{\nu\varepsilon} & P_{\nu\nu} \end{array}\right] \left[\frac{H_\epsilon^{\mathrm{T}}}{H_\nu^{\mathrm{T}}}\right] \end{aligned} \tag{9.138}$$

$$\left\{[H_\epsilon \quad | \quad H_\nu] \left[\begin{array}{c|c} P_{\epsilon\epsilon} & P_{\epsilon\nu} \\ \hline P_{\nu\epsilon} & P_{\nu\nu} \end{array}\right] \left[\frac{H_\epsilon^{\mathrm{T}}}{H_\nu^{\mathrm{T}}}\right] + R\right\}^{-1} \tag{9.139}$$

and the individual blocks

$$K_\epsilon = \{P_{\epsilon\epsilon}H_\epsilon^{\mathrm{T}} + P_{\epsilon v}H_v^{\mathrm{T}}\}C, \tag{9.140}$$

$$K_v = \{P_{v\epsilon}H_\epsilon^{\mathrm{T}} + P_{vv}H_v^{\mathrm{T}}\}C, \tag{9.141}$$

where the common factor

$$C = \left\{ [H_\epsilon \quad | \quad H_v] \begin{bmatrix} P_{\epsilon\epsilon} & P_{\epsilon v} \\ P_{v\epsilon} & P_{vv} \end{bmatrix} \begin{bmatrix} H_\epsilon^{\mathrm{T}} \\ \hline H_v^{\mathrm{T}} \end{bmatrix} + R \right\}^{-1} \tag{9.142}$$

$$= \{H_\epsilon P_{\epsilon\epsilon}H_\epsilon^{\mathrm{T}} + H_\epsilon P_{\epsilon v}H_v^{\mathrm{T}} + H_v P_{v\epsilon}H_\epsilon^{\mathrm{T}} + H_v P_{vv}H_v^{\mathrm{T}} + R\}^{-1}. \tag{9.143}$$

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.

9.6.3.2 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

$$\overline{K}_{\text{suboptimal}} = \begin{bmatrix} K_{\text{SK}} \\ 0 \end{bmatrix}, \tag{9.144}$$

where K_{SK} is the $n_\epsilon \times \ell$ 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) $K_v = 0$.

9.6.3.3 Suboptimal Observational Update The observational update equation for arbitrary gain K_k can be represented in the form

$$P_{k(+)} = (I_n - \overline{K}_k H_k)P_{k(-)}(I_n - \overline{K}_k H_k)^{\mathrm{T}} + \overline{K}_k R \overline{K}_k^{\mathrm{T}}, \tag{9.145}$$

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 that the suboptimal gain K_k has the partitioned form shown in Equation 9.144, the partitioned observational update equation for P will be

$$\begin{aligned}\begin{bmatrix} P_{\epsilon\epsilon,k(+)} & P_{\epsilon\upsilon,k(+)} \\ P_{\upsilon\epsilon,k(+)} & P_{\upsilon\upsilon,k(+)} \end{bmatrix} &= \left(\begin{bmatrix} I_{n_\epsilon} & 0 \\ 0 & I_{n_\upsilon} \end{bmatrix} - \begin{bmatrix} K_{\mathrm{SK},k} \\ 0 \end{bmatrix} \begin{bmatrix} H_{\epsilon,k} & | & H_{\upsilon,k} \end{bmatrix} \right) \\ &\quad \times \begin{bmatrix} P_{\epsilon\epsilon,k(-)} & P_{\epsilon\upsilon,k(-)} \\ P_{\upsilon\epsilon,k(-)} & P_{\upsilon\upsilon,k(-)} \end{bmatrix} \\ &\quad \times \left(\begin{bmatrix} I_{n_\epsilon} & 0 \\ 0 & I_{n_\upsilon} \end{bmatrix} - \begin{bmatrix} K_{\mathrm{SK},k} \\ 0 \end{bmatrix} \begin{bmatrix} H_{\epsilon,k} & H_{\upsilon,k} \end{bmatrix} \right)^{\mathrm{T}} \\ &\quad + \begin{bmatrix} K_{\mathrm{SK},k} a \\ 0 \end{bmatrix} R_k \begin{bmatrix} K_{\mathrm{SK},k}^{\mathrm{T}} & 0 \end{bmatrix}. \end{aligned} \tag{9.146}$$

The summed terms in parentheses can be combined into the following form for expansion:

$$\begin{aligned} &= \begin{bmatrix} I_{n_\epsilon} - K_{\mathrm{SK},k} H_{\epsilon,k} & -K_{\mathrm{SK},k} H_{\upsilon,k} \\ 0 & I_{n_\upsilon} \end{bmatrix} \times \begin{bmatrix} P_{\epsilon\epsilon,k(-)} & P_{\epsilon\upsilon,k(-)} \\ P_{\upsilon\epsilon,k(-)} & P_{\upsilon\upsilon,k(-)} \end{bmatrix} \\ &\quad \times \begin{bmatrix} I_{n_\epsilon} - H_{\epsilon,k}^{\mathrm{T}} K_{\mathrm{SK},k}^{\mathrm{T}} & 0 \\ -H_{\upsilon,k}^{\mathrm{T}} K_{\mathrm{SK},k}^{\mathrm{T}} & I_{n_\upsilon} \end{bmatrix} + \begin{bmatrix} K_{\mathrm{SK},k} R_k K_{\mathrm{SK},k}^{\mathrm{T}} & 0 \\ 0 & 0 \end{bmatrix}, \end{aligned} \tag{9.147}$$

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):

$$\begin{aligned} P_{\epsilon\epsilon,k(+)} &= \underbrace{(I_{n_\epsilon} - K_{\mathrm{SK},k} H_{\epsilon,k})}_{\mathcal{A}} P_{\epsilon\epsilon,k(-)} \underbrace{(I_{n_\epsilon} - K_{\mathrm{SK},k} H_{\epsilon,k})^{\mathrm{T}}}_{\mathcal{A}^{\mathrm{T}}} \\ &\quad - \underbrace{\overbrace{(I_{n_\epsilon} - K_{\mathrm{SK},k} H_{\epsilon,k})}^{\mathcal{A}} P_{\upsilon\epsilon,k(-)} H_{\upsilon,k}^{\mathrm{T}} K_{\mathrm{SK},k}^{\mathrm{T}}}_{\mathcal{B}} \\ &\quad - \overbrace{K_{\mathrm{SK},k} H_{\upsilon,k} \; P_{\epsilon\upsilon,k(-)} (I_{n_\epsilon} - K_{\mathrm{SK},k} H_{\epsilon,k})^{\mathrm{T}}}^{\mathcal{B}^{\mathrm{T}}} \\ &\quad + K_{\mathrm{SK},k} R_k K_{\mathrm{SK},k}^{\mathrm{T}}, \end{aligned} \tag{9.148}$$

$$P_{\epsilon\upsilon,k(+)} = \overbrace{(I_{n_\epsilon} - K_{\mathrm{SK},k} H_{\epsilon,k})}^{\mathcal{A}} P_{\epsilon\upsilon,k(-)} - K_{\mathrm{SK},k} H_{\upsilon,k} P_{\upsilon\upsilon,k(-)}, \tag{9.149}$$

$$P_{\upsilon\epsilon,k(+)} = P_{\epsilon\upsilon,k(+)}^{\mathrm{T}}, \tag{9.150}$$

$$P_{\upsilon\upsilon,k(+)} = P_{\upsilon\upsilon,k(-)}. \tag{9.151}$$

TABLE 9.2 Implementation Equations of Schmidt–Kalman Filter

Observational Update

$$
\begin{aligned}
\mathcal{C} &= \{H_{\epsilon k}(P_{\epsilon\epsilon k(-)}H_{\epsilon k}^{\mathrm{T}} + P_{\epsilon v k(-)}H_{vk}^{\mathrm{T}}) \\
&\quad + H_{vk}(P_{v\epsilon k(-)}H_{\epsilon k}^{\mathrm{T}} + P_{vvk(-)}H_{vk}^{\mathrm{T}}) + R_k\}^{-1} \\
K_{\mathrm{SK},k} &= \{P_{\epsilon\epsilon k-}H_{\epsilon k}^{\mathrm{T}} + P_{\epsilon v k(-)}H_{vk}^{\mathrm{T}}\}\mathcal{C} \\
\mathrm{x}_{\epsilon,k(+)} &= \mathrm{x}_{\epsilon,k-} + K_{\mathrm{S-K},k}\{\mathrm{z}_k - H_{\epsilon k}\mathrm{x}_{\epsilon,k(-)}\} \\
\mathcal{A} &= I_{n_\epsilon} - K_{\mathrm{SK},k}H_{\epsilon,k} \\
\mathcal{B} &= \mathcal{A}P_{v\epsilon,k(-)}H_{v,k}^{\mathrm{T}}K_{S-K,k}^{\mathrm{T}} \\
P_{\epsilon\epsilon,k(+)} &= \mathcal{A}P_{\epsilon\epsilon,k-}\mathcal{A}^{\mathrm{T}} - \mathcal{B} - \mathcal{B}^{\mathrm{T}} + K_{\mathrm{SK},k}R_kK_{\mathrm{SK},k}^{\mathrm{T}} \\
P_{\epsilon v,k(+)} &= \mathcal{A}P_{\epsilon v,k(-)} - K_{\mathrm{SK},k}H_{v,k}P_{vv,k(-)} \\
P_{v\epsilon,k(+)} &= P_{\epsilon v,k(+)}^{\mathrm{T}} \\
P_{vv,k(+)} &= P_{vv,k(-)}
\end{aligned}
$$

Temporal Update

$$
\begin{aligned}
\mathrm{x}_{\epsilon,k+1(-)} &= \Phi_{\epsilon k}\mathrm{x}_{\epsilon,k(+)} \\
P_{\epsilon\epsilon k+1(-)} &= \Phi_{\epsilon k}P_{\epsilon\epsilon k(+)}\Phi_{\epsilon k}^{\mathrm{T}} + Q_{\epsilon\epsilon} \\
P_{\epsilon v k+1(-)} &= \Phi_{\epsilon k}P_{\epsilon v k(+)}\Phi_{vk}^{\mathrm{T}} \\
P_{v\epsilon k+1(-)} &= P_{\epsilon v k+1(-)}^{\mathrm{T}} \\
P_{vvk+1(-)} &= \Phi_{vk}P_{vvk(+)}\Phi_{vk}^{\mathrm{T}} + Q_{vv}
\end{aligned}
$$

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.

9.6.4 Implementation Equations

We can now summarize just the essential equations from the derivation above, as listed in Table 9.2. These have been rearranged slightly to reuse intermediate results.

9.6.5 Computational Complexity

The purpose of the Schmidt–Kalman filter was to reduce the computational requirements over those required for the full Kalman filter. Although the equations appear to be more complicated, the dimensions of the matrices involved are smaller than 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 9.3 is a breakdown of the operations counts for implementing the equations in Table 9.2. The formulas (in angular 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_υ).

These complexity formulas are based on the matrix dimensions listed in Table 9.4.

A MATLAB implementation of the Schmidt–Kalman filter is in the m-file `KFvsSKF.m` on the Wiley web site.

9.7 MEMORY, THROUGHPUT, AND WORDLENGTH REQUIREMENTS

These may not be important issues for offline 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.

9.7.1 Wordlength Problems

9.7.1.1 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 7.

9.7.1.2 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 cases, but not all cases, the condition number of a matrix can be improved by rescaling. For example, the two covariance matrices

$$\begin{bmatrix} 1 & 0 \\ 0 & \epsilon^2 \end{bmatrix} \text{ and } \frac{1}{2}\begin{bmatrix} 1+\epsilon^2 & 1-\epsilon^2 \\ 1-\epsilon^2 & 1+\epsilon^2 \end{bmatrix}$$

have the same condition number ($1/\epsilon^2$), 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 $1/\epsilon$.

9.7.2 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

TABLE 9.3 Operations Counts for the Schmidt–Kalman Filter

Scalar Operation Counts for Matrix Operations	Totals by Rows
$\mathcal{C} = \{\overbrace{H_{\epsilon k}}^{\langle n_\epsilon \ell^2\rangle} \times (\underbrace{\overbrace{P_{\epsilon\epsilon k(-)}}^{\langle n_\epsilon^2 \ell\rangle} H_{\epsilon k}^{\mathrm{T}} + \overbrace{P_{\epsilon v k(-)}}^{\langle n_\epsilon n_v \ell\rangle} H_{vk}^{\mathrm{T}}}_{\langle \text{used again below}\rangle})$	$n_\epsilon \ell^2 + n_\epsilon^2 \ell + n_\epsilon n_v \ell$
$+ \overbrace{H_{vk}}^{\langle n_v \ell^2\rangle} \times (\overbrace{P_{v\epsilon k(-)}}^{\langle n_\epsilon n_v \ell\rangle} H_{\epsilon k}^{\mathrm{T}} + \overbrace{P_{vvk(-)}}^{\langle n_v^2 \ell\rangle} H_{vk}^{\mathrm{T}})$	$n_v \ell^2 + n_\epsilon n_v \ell + n_v^2 \ell$
$+R_k\}^{-1\langle \ell^3\rangle}$(matrix inverse)	ℓ^3
$K_{SK,k} = \{\overbrace{P_{\epsilon\epsilon k(-)} H_{\epsilon k}^{\mathrm{T}} + P_{\epsilon v k(-)} H_{vk}^{\mathrm{T}}}^{\langle \text{already computed above}\rangle}\} \overbrace{\times \mathcal{C}}^{\langle n_\epsilon \ell^2\rangle}$	$n_\epsilon \ell^2$
$\mathrm{x}_{\epsilon,k(+)} = \mathrm{x}_{\epsilon,k(-)} + \overbrace{K_{\mathrm{SK},k}}^{\langle n_\epsilon \ell\rangle} \times \overbrace{\{\mathrm{z}_k - H_{\epsilon k}\mathrm{x}_{\epsilon,k(-)}\}}^{\langle n_\epsilon \ell\rangle}$	$2n_\epsilon \ell$
$\mathcal{A} = I_{n_\epsilon} - \overbrace{K_{\mathrm{SK},k}}^{\langle n_\epsilon^2 \ell\rangle} H_{\epsilon,k}$	$n_\epsilon^2 \ell$
$\mathcal{B} = \underbrace{\overbrace{\mathcal{A}}^{\langle n_\epsilon^2 n_v\rangle} \times P_{\epsilon v,k(-)}}_{\langle n_\epsilon^2 n_v\rangle} \times \overbrace{H_{v,k}^{\mathrm{T}} \times K_{\mathrm{SK},k}^{\mathrm{T}}}^{\langle n_\epsilon n_v \ell\rangle}$	$2n_\epsilon^2 n_v + n_\epsilon n_v \ell$
$P_{\epsilon\epsilon,k(+)} = \mathcal{A} P_{\epsilon\epsilon,k(-)} \mathcal{A}^{\mathrm{T}} - \mathcal{B} - \mathcal{B}^{\mathrm{T}}$	$\frac{3}{2}n_\epsilon^3 + \frac{1}{2}n_\epsilon^2$
$+\underbrace{K_{\mathrm{SK},k}\overbrace{[H_{vk}, P_{vv,k(-)} H_{v,k}^{\mathrm{T}} + R_k]}^{\langle n_\epsilon^2 n_v\rangle} K_{\mathrm{SK},k}^{\mathrm{T}}}_{\langle \frac{1}{2}n_\epsilon^2 \ell + n_\epsilon \ell^2 + \frac{1}{2}n_\epsilon \ell\rangle}$	$\frac{3}{2}n_v^3 + n_v^2 \ell + \frac{1}{2}n_v \ell^2 + \frac{1}{2}n_v^2 + \frac{1}{2}n_v \ell$
$P_{\epsilon v,k(+)} = \overbrace{\mathcal{A} \times P_{\epsilon v,k(-)}}^{\langle n_\epsilon^2 n_v\rangle} - \overbrace{K_{\mathrm{SK},k} \times H_{v,k} \times P_{vv,k(-)}}^{\langle n_\epsilon n_v \ell\rangle \quad \langle n_\epsilon n_v^2\rangle}$	$n_\epsilon^2 n_v + n_\epsilon n_v \ell + n_\epsilon n_v^2$
$P_{ve,k(+)} = P_{\epsilon v,k(+)}^{\mathrm{T}}$	0
$P_{vv,k(+)} = P_{vv,k(-)}$	0
Total for Observational Update	$3n_\epsilon \ell^2 + \frac{5}{2}n_\epsilon^2 \ell + 4n_\epsilon n_v \ell + \frac{3}{2}n_v \ell^2 + 2n_v^2 \ell + \ell^3 + \frac{5}{2}n_\epsilon \ell + 3n_\epsilon^2 n_v + \frac{1}{2}n_v \ell + n_\epsilon n_v^2$
Temporal Update	
$\hat{\mathbf{x}}_{\epsilon,k+1(-)} = \Phi_{\epsilon k} \hat{\mathbf{x}}_{\epsilon,k(+)}$	n_ϵ^2
$P_{\epsilon\epsilon k+1(-)} = \Phi_{\epsilon k} P_{\epsilon\epsilon k(+)} \Phi_{\epsilon k}^{\mathrm{T}} + Q_{\epsilon\epsilon}$	$\frac{3}{2}n_\epsilon^3 + \frac{1}{2}n_\epsilon^2$
$P_{\epsilon v k+1(-)} = \Phi_{\epsilon k} P_{\epsilon v k(+)} \Phi_{vk}^{\mathrm{T}}$	$n_\epsilon n_v^2 + n_v n_\epsilon^2$
$P_{ve k+1(-)} = P_{\epsilon v k+1-}^{\mathrm{T}}$	0
$P_{vv k+1(-)} = \Phi_{vk} P_{vv k(+)} \Phi_{vk}^{\mathrm{T}} + Q_{vv}$	$\frac{3}{2}n_v^3 + \frac{1}{2}n_v^2$
Total for Temporal Update	$\frac{3}{2}n_\epsilon^2 + \frac{3}{2}n_\epsilon^3 + n_\epsilon n_v^2 + n_\epsilon^2 n_v + \frac{3}{2}n_v^3 + \frac{1}{2}n_v^2$
Total for Schmidt–Kalman Filter	$3n_\epsilon \ell^2 + \frac{5}{2}n_\epsilon^2 \ell + 4n_\epsilon n_v \ell + \frac{3}{2}n_v \ell^2 + 2n_v^2 \ell + \ell^3 + \frac{5}{2}n_\epsilon \ell + 4n_\epsilon^2 n_v + \frac{3}{2}n_\epsilon^3 + \frac{3}{2}n_\epsilon^2 + \frac{1}{2}n_v \ell + \frac{3}{2}n_v^3 + \frac{1}{2}n_v^2 + \frac{1}{2}n_v \ell + 2n_\epsilon n_v^2$

TABLE 9.4 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
$P_{\epsilon\epsilon}$	n_ϵ	n_ϵ
$P_{\epsilon v}$	n_ϵ	n_v
$P_{v\epsilon}$	n_v	n_ϵ
P_{vv}	n_v	n_v
$Q_{\epsilon\epsilon}$	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
$\hat{\mathbf{x}}_\epsilon$	n_ϵ	1
$\mathbf{z}$	ℓ	1

filters. A few of these techniques have been mentioned in Chapter 7, 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.

9.7.2.1 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 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 is very much dependent on the problem size. We present here some general formulas for this dependence.

9.7.2.2 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.

TABLE 9.5 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	
	Φx	n	$+2n$

*In units of floating-point data words.

9.7.2.3 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 9.5 and plotted in Figure 9.26. This is the Kalman filter implementation diagrammed in Figure 7.2, which reuses some partial results. 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 $\Phi P, \Phi$, and GQG^{T}, the operation $P \leftarrow \Phi P \Phi^{\mathrm{T}} + GQG^{\mathrm{T}}$ is performed in place;

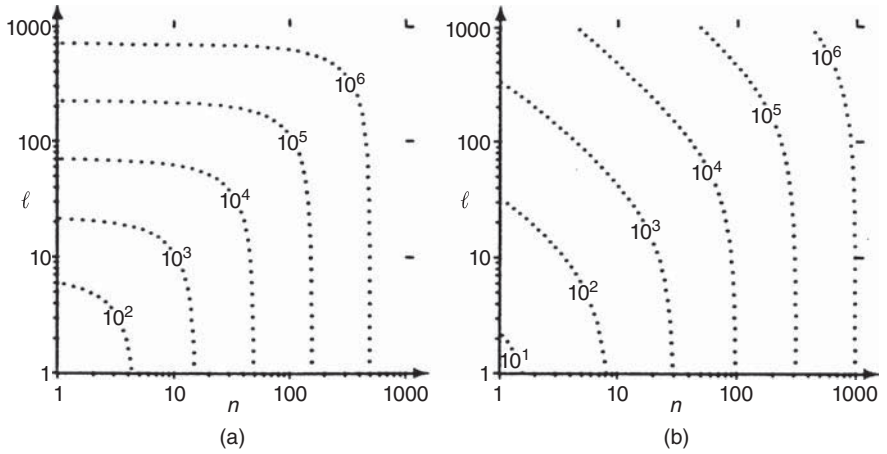


Figure 9.26 Conventional filter memory requirements (in words) versus state dimension (n) and measurement dimension (l). (a) Complete implementation and (b) without gain calculations.

- 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 9.26 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.

9.7.2.4 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 be primarily from index computations, which are not the standard ones used by optimizing compilers. Table 9.6 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.

TABLE 9.6 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; and k is the corresponding index of a one-dimensional array.
†In units of data words.

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 $k(i, j)$.

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 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

$$F = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}, \tag{9.152}$$

$$\Phi(t) = e^{Ft} \tag{9.153}$$

$$= \begin{bmatrix} 1 & t & \frac{1}{2}t^2 & \cdots & \frac{1}{(n-1)!}t^{n-1} \\ 0 & 1 & t & \cdots & \frac{1}{(n-2)!}t^{n-2} \\ 0 & 0 & 1 & \cdots & \frac{1}{(n-3)!}t^{n-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}, \tag{9.154}$$

where t is the discrete-time interval. The following algorithm computes the product $M = \Phi P$, with Φ as given in Equation 9.154, 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.

9.7.3 Throughput, Processor Speed, and Computational Complexity

9.7.3.1 Computational Complexity and 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 7. These are the *maximum* computational complexities of the implementation methods listed in Chapter 7. The computational complexity of an application can be made smaller if it includes sparse matrix operations that can be implemented more efficiently.

9.7.3.2 Conventional Kalman Filter The maximum computational complexity of the conventional Kalman filter implementation is plotted versus problem size in Figure 9.27. This implementation uses all the shortcuts listed in Section 9.7.2 and also

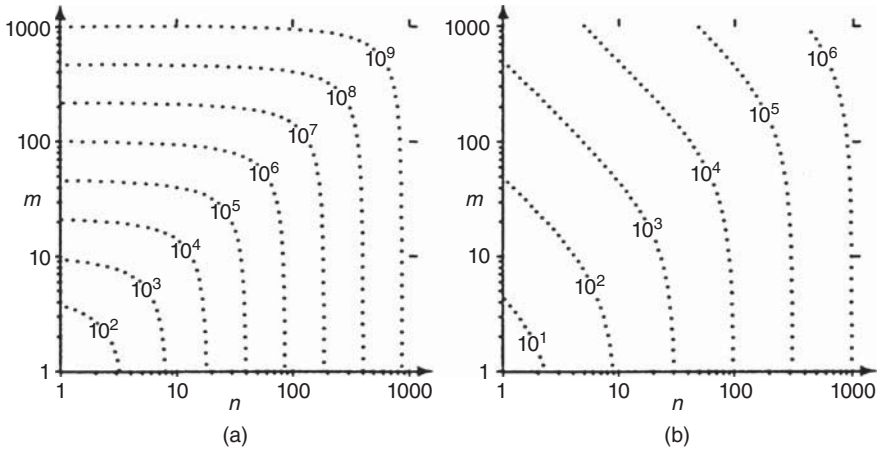


Figure 9.27 Contour plots of computational complexity (in flops per measurement) of the Kalman filter as a function of state dimension (n) and measurement dimension (m). (a) Complete and (b) without the Riccati equation.

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 offline, as in a gain-scheduled or steady-state implementation.

9.7.3.3 Bierman–Thornton Square-Root Implementation The corresponding dependence of computational complexity on problem size for the *UD* filter implementation is plotted in Figure 9.28(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 that Q and R are already diagonal are displayed in Figure 9.28(b).

9.7.4 Programming Cost versus Runtime Cost

The computational complexity issue in Kalman filtering is usually driven by the need to execute in real time. The computational complexity grows so fast with the problem size that it will overwhelm even the fastest processors for sufficiently large system models. For that reason, the issue of computational complexity is one 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 the straightforward programming solutions to produce acceptable results.

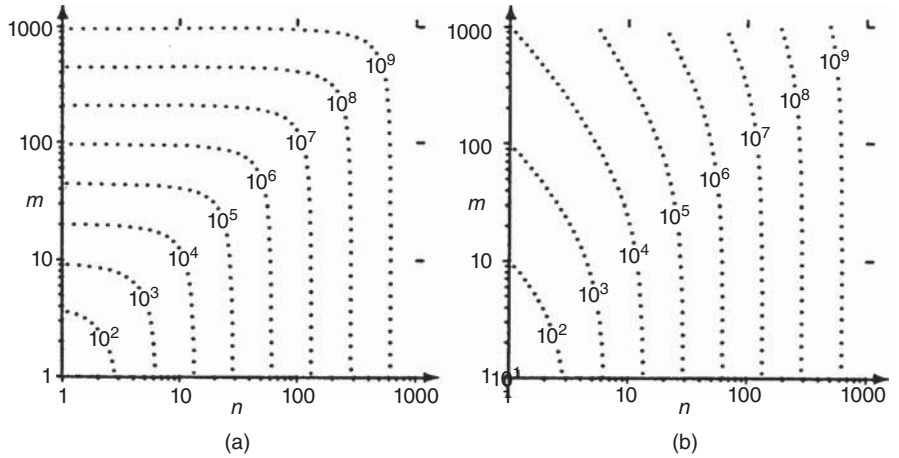


Figure 9.28 Contour plots of computational complexity (in flops per measurement) of the Bierman–Thornton implementation as a function of state dimension (n) and measurement dimension (m). (a) Full Q and R matrices and (b) diagonal Q and R matrices.

9.8 WAYS TO REDUCE COMPUTATIONAL REQUIREMENTS

9.8.1 Reducing Complexities of Matrix Products

9.8.1.1 Products of Two Matrices The number of flops required to compute the product of general $\ell \times m$ and $m \times n$ matrices is $\ell m^2 n$. This figure can be reduced 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 for products involving diagonal or triangular factors. They should always be exploited whenever H or Φ is a sparse matrix.

9.8.1.2 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

$$M_1 \times (M_2 \times M_3) = (M_1 \times M_2) \times M_3 \tag{9.155}$$

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 9.7. The number of flops depends on the order of multiplication, being $n_2(n_3^3 + n_1 n_2)n_4$ in one case and $n_1(n_2^2 + n_3 n_4)n_3$ in the other case. The implementation $M_1 \times (M_2 \times M_3)$ is favored if

TABLE 9.7 Computational Complexities of 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	$n_2 n_3^2 n_4$	$n_1 n_2^2 n_3$
Second multiply	$n_1 n_2^2 n_4$	$n_1 n_3^2 n_4$
Total	$n_2(n_3^3 + n_1 n_2)n_4$	$n_1(n_2^2 + n_3 n_4)n_3$

$n_1 n_2^2(n_4 - n_3) < (n_1 - n_2)n_3^2 n_4$, 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 7.6.1.4).

9.8.2 Offline versus Online 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 "online" and those that can be performed "offline" (i.e., carried out beforehand, with the results stored in memory, including bulk media, such as magnetic tape or CDROM, and read back in real time[7]). The online 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 5 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 offline. It 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 of greater efficiency (but less generality) are discussed in the following subsections. Methods for performance analysis of these suboptimal estimation methods are discussed in Section 9.5.

9.8.3 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

[7] In assessing the real-time implementation requirements, one must trade off the time to read these prestored values versus the time required to compute them. In some cases, the read times may exceed the computation times.

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 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 in the interior of the stage in which they are used or weighted averages of all the exact values over the range.

The performance trade-off between the decreased storage requirements (for using fewer values of the gains) and the increased approximation error (due to differences between the optimal gains and the scheduled gains) can be analyzed by simulation.

9.8.4 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.

There are several methods for solving the steady-state matrix Riccati equation in the following subsections. One of these (the doubling method) is based on the linearization method for the Riccati equation presented in Chapter 5. Theoretically, it converges exponentially faster than the serial iteration method. In practice, however, convergence can stall (due to numerical problems) before an accurate solution is attained. However, it can still be used to obtain a good starting estimate for the Newton–Raphson method (described in Chapter 5).

9.8.4.1 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 5.2. As in the continuous case, the asymptotic solution should equal the solution of the *steady-state equation*:

$$P_\infty = \Phi[P_\infty - P_\infty H^{\mathrm{T}}(HP_\infty H^{\mathrm{T}} + R)^{-1} HP_\infty]\Phi^{\mathrm{T}} + Q, \tag{9.156}$$

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

$$\Psi = \begin{bmatrix} (\Phi + Q\Phi^{-\mathrm{T}} HR^{-1}H^{\mathrm{T}}) & Q\Phi^{-\mathrm{T}} \\ \Phi^{-\mathrm{T}} R^{-1} & \Phi^{-\mathrm{T}} \end{bmatrix}. \tag{9.157}$$

The pth squaring of Ψ will then yield Ψ^{2p} and the solution

$$P_{2p(-)} = A_{2p}B_{2p}^{-1} \tag{9.158}$$

for

$$\begin{bmatrix} A_{2p} \\ B_{2p} \end{bmatrix} = \Psi^{2p} \begin{bmatrix} A_0 \\ B_0 \end{bmatrix}. \tag{9.159}$$

Davison–Maki–Friedlander–Kailath Squaring Algorithm Note that if one expresses Ψ^{2^N} in symbolic form as

$$\Psi^{2^N} = \begin{bmatrix} \mathcal{A}_N^{\mathrm{T}} + \mathcal{C}_N\mathcal{A}_N^{-1}\mathcal{B}_N & \mathcal{C}_N\mathcal{A}_N^{-1} \\ \mathcal{A}_N^{-1}\mathcal{B}_N & \mathcal{A}_N^{-1} \end{bmatrix}, \tag{9.160}$$

then its square can be put in the form

$$\Psi^{2^{N+1}} = \begin{bmatrix} \mathcal{A}_N^{\mathrm{T}} + \mathcal{C}_N\mathcal{A}_N^{-1}\mathcal{B}_N & \mathcal{C}_N\mathcal{A}_N^{-1} \\ \mathcal{A}_N^{-1}\mathcal{B}_N & \mathcal{A}_N^{-1} \end{bmatrix}^2 \tag{9.161}$$

$$= \begin{bmatrix} \mathcal{A}_{N+1}^{\mathrm{T}} + \mathcal{C}_{N+1}\mathcal{A}_{N+1}^{-1}\mathcal{B}_{N+1} & \mathcal{C}_{N+1}\mathcal{A}_{N+1}^{-1} \\ \mathcal{A}_{N+1}^{-1}\mathcal{B}_{N+1} & \mathcal{A}_{N+1}^{-1} \end{bmatrix}, \tag{9.162}$$

$$\mathcal{A}_{N+1} = \mathcal{A}_N(I + \mathcal{B}_N\mathcal{C}_N)^{-1}\mathcal{A}_N, \tag{9.163}$$

$$\mathcal{B}_{N+1} = \mathcal{B}_N + \mathcal{A}_N(I + \mathcal{B}_N\mathcal{C}_N)^{-1}\mathcal{B}_N\mathcal{A}_N^{\mathrm{T}}, \tag{9.164}$$

$$\mathcal{C}_{N+1} = \mathcal{C}_N + \mathcal{A}_N^{\mathrm{T}}\mathcal{C}_N(I + \mathcal{B}_N\mathcal{C}_N)^{-1}\mathcal{A}_N. \tag{9.165}$$

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 9.157:

$$\mathcal{A}_0 = \Phi^{\mathrm{T}}, \tag{9.166}$$

$$\mathcal{B}_0 = H^{\mathrm{T}}R^{-1}H, \tag{9.167}$$

$$\mathcal{C}_0 = Q. \tag{9.168}$$

9.8.4.2 Initial Conditions If the initial value of the Riccati equation is with $P_0 = 0$, the zero matrix, it can be represented by $P_0 = A_0B_0^{-1}$ for $A_0 = 0$ and any nonsingular

TABLE 9.8 Davison–Maki–Friedlander–Kailath Squaring Algorithm

Initialization	Iteration (N times)
$\mathcal{A} = \Phi^{\mathrm{T}}$	$\mathcal{A} \leftarrow \mathcal{A}(I + \mathcal{B}C)^{-1}\mathcal{A}^{\mathrm{T}}$
$\mathcal{B} = H^{\mathrm{T}}R^{-1}H$	$\mathcal{B} \leftarrow \mathcal{B} + \mathcal{A}(I + \mathcal{B}C)^{-1}\mathcal{B}\mathcal{A}^{\mathrm{T}}$
$C = Q \ = P_1$	$C \leftarrow C + \mathcal{A}^{\mathrm{T}}C(I + \mathcal{B}C)^{-1}\mathcal{A}$
Termination	
$P_{2^N} = C$	

B_0. Then the Nth iterate of the doubling algorithm will yield

$$\begin{bmatrix} A_{2^N} \\ B_{2^N} \end{bmatrix} = \Psi^{2^N} \begin{bmatrix} A_0 \\ B_0 \end{bmatrix} \tag{9.169}$$

$$= \begin{bmatrix} \mathcal{A}_N^{\mathrm{T}} + C_N\mathcal{A}_N^{-1}\mathcal{B}_N & C_N\mathcal{A}_N^{-1} \\ \mathcal{A}_N^{-1}\mathcal{B}_N & \mathcal{A}_N^{-1} \end{bmatrix} \begin{bmatrix} 0 \\ B_1 \end{bmatrix} \tag{9.170}$$

$$= \begin{bmatrix} C_N\mathcal{A}_N^{-1}\mathcal{B}_1 \\ \mathcal{A}_N^{-1}B_1 \end{bmatrix}, \tag{9.171}$$

$$P_{2^N} = A_{2^N}B_{2^N}^{-1} \tag{9.172}$$

$$= C_N\mathcal{A}_N^{-1}B_1(\mathcal{A}_N^{-1}B_1)^{-1} \tag{9.173}$$

$$= C_N. \tag{9.174}$$

That is, after the Nth squaring step, the submatrix

$$C_N = P_{2^N}. \tag{9.175}$$

The resulting algorithm is summarized in Table 9.8. 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[8]. An array allocation scheme for performing the squaring algorithm using only six $n \times n$ arrays is shown in Figure 9.29.

9.8.4.3 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 C. Consequently, if $\|\mathcal{A}_N\| \ll 1$, then the computed values of $\mathcal{B}_N$ and 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 those stall situations, the doubling algorithm is still an efficient method for getting an approximate nonnegative-definite solution.

[8]The matrices $\mathcal{B}_N$ and C_N and the matrix products $(I + \mathcal{B}_N C_N)^{-1}B_N$ and $C_N(I + \mathcal{B}_N C_N)^{-1}$ 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 requires about one-fourth less memory than straightforward squaring, also.

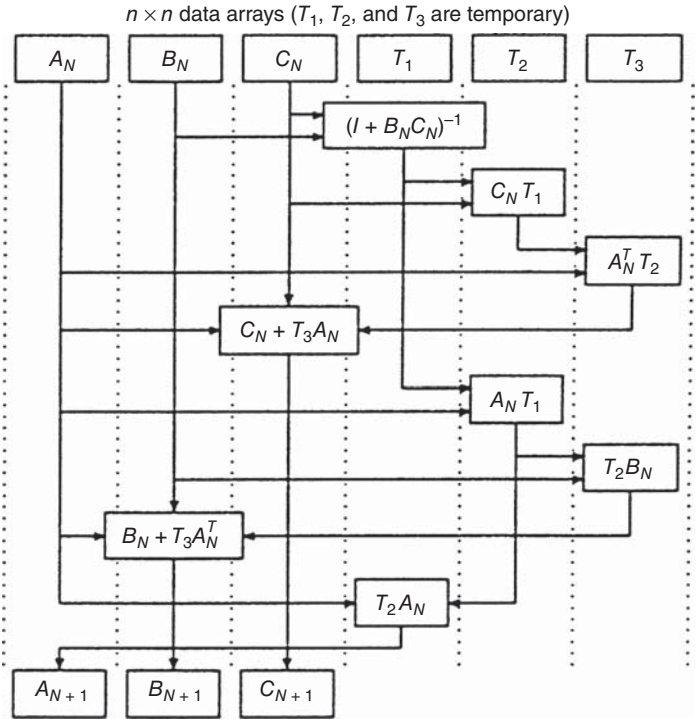


Figure 9.29 Data array usage for doubling algorithm.

9.9 ERROR BUDGETS AND SENSITIVITY ANALYSIS

9.9.1 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.

9.9.2 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 development and testing.
   (c) Reducing other system-level performance attributes, such as cost, size, weight, and power. This also includes such practices as suboptimal filtering methods to reduce computational requirements.

9.9.2.1 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 9.30. The rows represent the contributions of major sensor subsystems, and the columns represent their contributions to each of the multiple system-level error

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 9.30 Error budget breakdown.

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.

9.9.3 Error Sensitivity Analysis and Budgeting

A 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 or 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.

9.9.3.1 Dual-state System Modeling 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 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 9.31. 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 one can 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 only accurate for small changes in the budget entries.

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

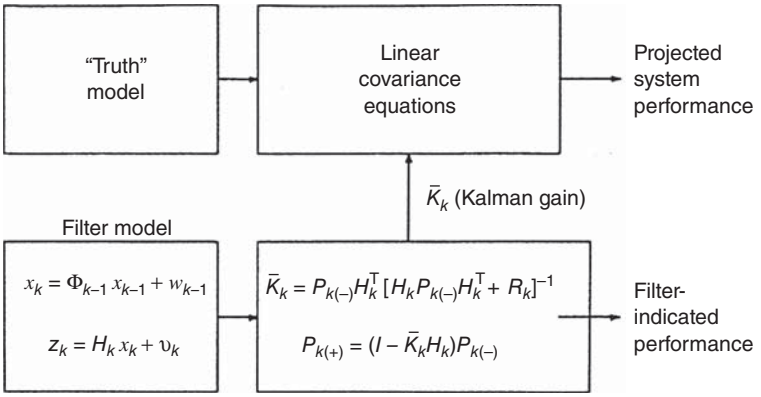


Figure 9.31 Error budgeting model.

"finessing" these subsystem-level error allocations to arrive at a more reasonable distribution.

9.9.4 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, very many Monte Carlo runs must be made to obtain reasonable confidence in the results.

Monte Carlo analysis has certain advantages over covariance analysis, however. The 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 will tend to show up under these test conditions.

9.10 OPTIMIZING MEASUREMENT SELECTION POLICIES

9.10.1 Measurement Selection Problem

9.10.1.1 Relation to Kalman Filtering and Error Budgeting We 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 [7].

9.10.1.2 Optimization with Respect to 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 $\hat{x} - x$:

$$\mathcal{L} = \sum_{\ell=1}^{N} \|A_\ell(\hat{x}_{\ell(+)} - x_\ell)\|^2, \tag{9.176}$$

where the given matrices A_ℓ transform the estimation errors to 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 transformation. 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.

9.10.2 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[9] function on the selection of measurements.

9.10.2.1 Parameterizing the Possible Measurements As far as the Kalman filter is concerned, a measurement is characterized by H (its measurement sensitivity matrix)

[9]The term *risk* is here used to mean the expected loss.

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.

9.10.2.2 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

$$\mathcal{R}_k(P_k(+)) = E\left\{\sum_{\ell=k}^{N} \|A_\ell(\hat{x}_{\ell(+)} - x_\ell)\|^2\right\}, \tag{9.177}$$

where $P_k(+)$ will depend on the choice for the kth measurement and, for $k < \ell \leq N$,

$$\hat{x}_{\ell+1(+)} = \hat{x}_{\ell+1(-)}, \tag{9.178}$$

$$\hat{x}_{\ell+1(+)} - x_{\ell+1} = \Phi_\ell(\hat{x}_{\ell(+)} - x_\ell) - w_\ell, \tag{9.179}$$

so long as no additional measurements are used.

9.10.2.3 Marginal Risk Function Before proceeding further 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 9.1 For $0 \leq k \leq N$, the risk function defined by Equation 9.177 can be represented in the form

$$\mathcal{R}_k(P_k) = \text{trace}\,\{P_k W_k + V_k\}, \tag{9.180}$$

where

$$W_N = A_N^{\mathrm{T}} A_N, \tag{9.181}$$

$$V_N = 0, \tag{9.182}$$

and, for $\ell < N$,

$$W_\ell = \Phi_\ell^{\mathrm{T}} W_{\ell+1} \Phi_\ell + A_\ell^{\mathrm{T}} A_\ell, \tag{9.183}$$

$$V_\ell = Q_\ell W_{\ell+1} + V_{\ell+1}. \tag{9.184}$$

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 from Equations 9.181 and 9.182 into Equation 9.180, and substituting N for k, one can obtain the following sequence of equalities:

$$\begin{aligned}
\mathcal{R}_N(P_N) &= \mathrm{trace}\{P_N W_N + V_N\} \\
&= \mathrm{trace}\{P_N A_N^{\mathrm{T}} A_N + 0_{n\times n}\} \\
&= \mathrm{trace}\{A_N P_N A_N^{\mathrm{T}}\} \\
&= \mathrm{trace}\{A_N E\langle(\hat{x}_N - x_N)(\hat{x}_N - x_N)^{\mathrm{T}}\rangle A_N^{\mathrm{T}}\} \\
&= \mathrm{trace}\{\mathrm{E}\langle A_N(\hat{x}_N - x_N)(\hat{x}_N - x_N)^{\mathrm{T}} A_N^{\mathrm{T}}\rangle\} \\
&= \mathrm{trace}\{\mathrm{E}\langle[A_N(\hat{x}_N - x_N)][A_N(\hat{x}_N - x_N)]^{\mathrm{T}}\rangle\} \\
&= \mathrm{trace}\{\mathrm{E}\langle[A_N(\hat{x}_N - x_N)]^{\mathrm{T}}[A_N(\hat{x}_N - x_N)]\rangle\} \\
&= \mathrm{E}\langle\|A_N(\hat{x}_N - x_N)\|^2\rangle.
\end{aligned}$$

The first of these is Equation 9.180 for $k = N$, and the last is Equation 9.177 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 9.180 is equivalent to Equation 9.177 for $k = \ell + 1$ and seek to prove from that it must also be the case for $k = \ell$. Then start with Equation 9.177, noting that it can be written in the form

$$\begin{aligned}
\mathcal{R}_\ell(P_\ell) &= \mathcal{R}_{\ell+1}(P_{\ell+1}) + \mathrm{E}\langle\|A_\ell(\hat{x}_\ell - x_\ell)\|^2\rangle \\
&= \mathcal{R}_{\ell+1}(P_{\ell+1}) + \mathrm{trace}\{\mathrm{E}\langle\|A_\ell(\hat{x}_\ell - x_\ell)\|^2\rangle\}
\end{aligned}$$

$$
\begin{aligned}
&= \mathcal{R}_{\ell+1}(P_{\ell+1}) + \text{trace}\{\text{E}\langle[A_\ell(\hat{x}_\ell - x_\ell)]^{\text{T}}[A_\ell(\hat{x}_\ell - x_\ell)]\rangle\} \\
&= \mathcal{R}_{\ell+1}(P_{\ell+1}) + \text{trace}\{\text{E}\langle[A_\ell(\hat{x}_\ell - x_\ell)][A_\ell(\hat{x}_\ell - x_\ell)]^{\text{T}}\rangle\} \\
&= \mathcal{R}_{\ell+1}(P_{\ell+1}) + \text{trace}\{A_\ell \text{E}\langle(\hat{x}_\ell - x_\ell)(\hat{x}_\ell - x_\ell)^{\text{T}}\rangle A_\ell^{\text{T}}\} \\
&= \mathcal{R}_{\ell+1}(P_{\ell+1}) + \text{trace}\{A_\ell P_\ell A_\ell^{\text{T}}\} \\
&= \mathcal{R}_{\ell+1}(P_{\ell+1}) + \text{trace}\{P_\ell A_\ell^{\text{T}} A_\ell\}.
\end{aligned}
$$

Now one can use the assumption that Equation 9.180 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:

$$
\begin{aligned}
\mathcal{R}_\ell(P_\ell) &= \text{trace}\{P_{\ell+1}\ W_{\ell+1} + V_{\ell+1}\} + \text{trace}\{P_\ell A_\ell^{\text{T}} A_\ell\} \\
&= \text{trace}\{P_{\ell+1}\ W_{\ell+1} + V_{\ell+1} + P_\ell A_\ell^{\text{T}} A_\ell\} \\
&= \text{trace}\{[\Phi_\ell P_\ell \Phi_\ell^{\text{T}} + Q_\ell] W_{\ell+1} + V_{\ell+1} + P_\ell A_\ell^{\text{T}} A_\ell\} \\
&= \text{trace}\{\Phi_\ell P_\ell \Phi_\ell^{\text{T}} W_{\ell+1} + Q_\ell W_{\ell+1} + V_{\ell+1} + P_\ell A_\ell^{\text{T}} A_\ell\} \\
&= \text{trace}\{P_\ell \Phi_\ell^{\text{T}} W_{\ell+1} \Phi_\ell + Q_\ell W_{\ell+1} + V_{\ell+1} + P_\ell A_\ell^{\text{T}} A_\ell\} \\
&= \text{trace}\{P_\ell [\Phi_\ell^{\text{T}} W_{\ell+1} \Phi_\ell + A_\ell^{\text{T}} A_\ell] + [Q_\ell W_{\ell+1} + V_{\ell+1}]\} \\
&= \text{trace}\{P_\ell [W_\ell] + [V_\ell]\},
\end{aligned}
$$

where the Equations 9.183 and 9.184 were used in the last substitution. The last equation is Equation 9.180 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 the evaluation of the complete marginal risk function.

9.10.2.4 Marginal Benefit 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:

$$\mathcal{B}(H,R) = \mathcal{R}_k(P_{k(-)}) - \mathcal{R}_k(P_{k(+)}) \tag{9.185}$$

$$= \text{trace}\{[P_{k(-)} - P_{k(+)}]W_k\} \tag{9.186}$$

$$= \text{trace}\{[P_{k(-)}H^{\mathrm{T}}(HP_{k(-)}H^{\mathrm{T}} + R)^{-1}HP_{k(-)}]W_k\} \tag{9.187}$$

$$= \text{trace}\{(HP_{k(-)}H^{\mathrm{T}} + R)^{-1}HP_{k(-)}W_kP_{k(-)}H^{\mathrm{T}}\}. \tag{9.188}$$

This last formula is in a form useful for implementation.

9.10.3 Solution Algorithm for Maximum Marginal Benefit

1. Compute the matrices W_ℓ using the formulas given by Equations 9.181 and 9.183.
2. Select the measurements in temporal order: for $k = 0, 1, 2, 3, \ldots, N$:
   (a) For each possible measurement, using Equation 9.188, 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.

9.10.4 Computational Complexity

9.10.4.1 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 $\mathcal{O}(Nn^2(p+n))$.

9.10.4.2 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 9.9. 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[10] 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 $\mathcal{O}(N\mu\ell(\ell^2 + n^2))$.

[10]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 9.9 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.

9.11 SUMMARY

This chapter 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 wordlength 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 inertial navigation system (INS)-aided GPS navigator.

PROBLEMS

9.1 Show that the final value of the risk obtained by the marginal optimization technique of Section 9.10 will equal the initial risk minus the sum of the marginal benefits of the measurements selected.

9.2 Develop the equations for the dual-state error propagation by substituting Equations 9.107 and 9.108 into Equations 9.111 and 9.112 using Equation 9.114, explicitly.

9.3 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

$$\dot{x}^F = w^F, \quad z_K = x_k^F + v_k.$$

9.4 Derive the results of Example 7.4.

9.5 Prove that $\text{cov}[\tilde{x}_k^s]$ depends upon $\text{cov}(x_k^s)$.

9.6 Rework Problem 5.7 for the UDU^{T} formulation and compare your results with those of Problem 5.7.

9.7 Rework Problem 5.8 for the UDU^{T} formulation and compare your results with those of Problem 5.8.

9.8 Solve Problem 5.7 using the Schmidt–Kalman filter (Section 9.6) and compare the results with Example 5.8.

REFERENCES

[1] T. Kailath, "A general likelihood-ratio formula for random signals in Gaussian noise," *IEEE Transactions on Information Theory*, Vol. 15, No. 3, pp. 350–361, 1969.

[2] 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.

[3] 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.

[4] P. S. Maybeck, *Stochastic Models, Estimation, and Control*, Vol. 1, Academic Press, New York, 1979.

[5] S. F. Schmidt, "Applications of state-space methods to navigation problems," in *Advances in Control Systems*, Vol. 3 (C. T. Leondes, ed.), Academic Press, New York, pp. 293–340, 1966.

[6] R. G. Brown and P. Y. C. Hwang, *Introduction to Random Signals and Applied Kalman Filtering*, 4th ed., John Wiley & Sons, Inc., New York, 2012.

[7] A. Andrews, "Marginal optimization of observation schedules," *AIAA Journal of Guidance and Control*, Vol. 5, pp. 95–96, 1982.

10

APPLICATIONS TO NAVIGATION

> The acquisition of the knowledge of navigation has a strange effect on the minds of men.
>
> —Jack London
> The Cruise of the Snark, Macmillan, 1911.

10.1 CHAPTER FOCUS

It is hard to imagine an application for Kalman filtering more fruitful than navigation. Before Kalman filtering, navigation was practiced only by technical specialists with years if training. Today, it is available as a commodity service for the consumer, requiring no more skill than using a smartphone.

When the Kalman filter was published in 1960, it would find immediate applications in many military systems, including inertial and satellite navigation. It would play a major role in the development of satellite navigation, in particular, and in the integration of inertial and satellite navigation systems.

This has spawned a vast area of technology that had been largely classified until the late twentieth century but has since seen many successful commercial applications, illustrating how capabilities of Kalman filtering can be matched to the demands of very complex estimation problems.

The focus here will be on the Kalman filter architectures illustrated in Figure 10.1, and particularly on the contents of the dashed boxes within the two boxes labeled "Kalman filter." We shall not delve too deeply into what happens in the other boxes,

Kalman Filtering: Theory and Practice Using MATLAB®, Fourth Edition.
Mohinder S. Grewal and Angus P. Andrews.
© 2015 John Wiley & Sons, Inc. Published 2015 by John Wiley & Sons, Inc.

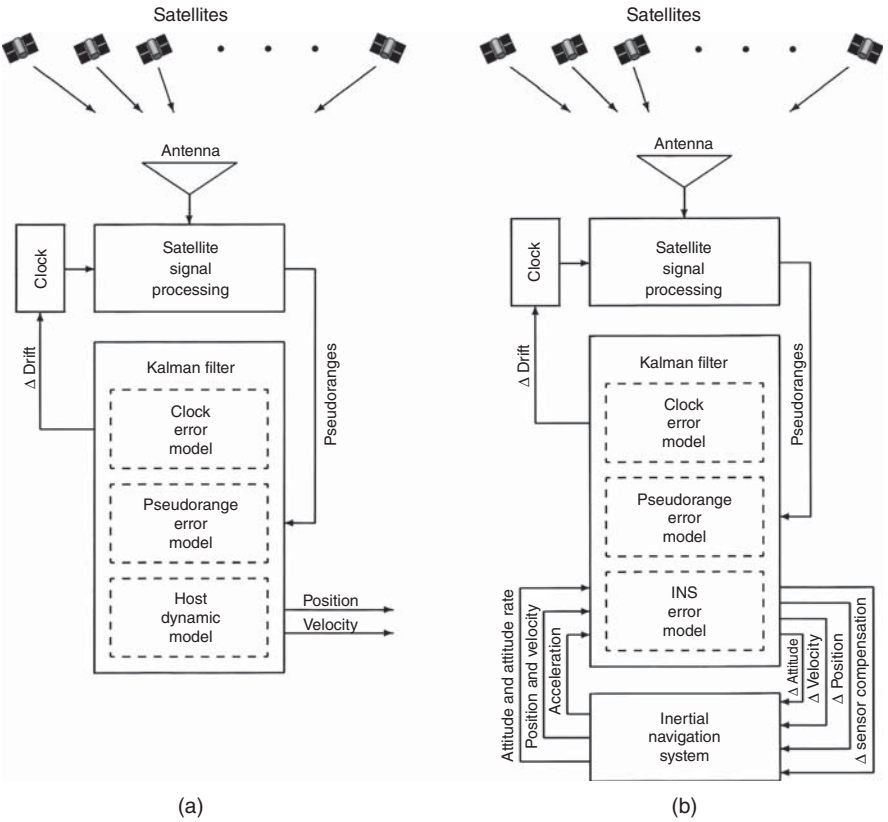


Figure 10.1 Kalman filter architectures for (a) GNSS and (b) GNSS/INS.

but we will derive and demonstrate Kalman filter models to show how it is done. The accompanying software also generates results beyond what can be shown here, just to show the enormous capabilities of the Kalman filtering approach to navigation.

10.2 NAVIGATION OVERVIEW

10.2.1 The Navigation Problem

The purpose of navigation is to direct the movement of a vehicle (wheeled, legged, waterborne, airborne, or spaceborne) 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.).

The solution to the navigation problem generally requires observations or measurements of some kind and being able to use that information 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 human kind.
2. Dead reckoning, which relies on knowing where you started from, plus some form of heading information (e.g., magnetic compass) and some estimate of distance traveled. Its implementation originally involved plotting on charts using drafting tools, but this operation is now done in software.
3. Celestial navigation, using time and the angles between local vertical and known celestial objects (e.g., sun, moon, planets, stars) to estimate orientation, latitude, and longitude. It depends on clear viewing conditions, and generally requires specialized instruments such as sextants and chronometers.
4. Radio navigation, which relies on radio-frequency sources with known signal characteristics and 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 is about applications of Kalman filtering in solving the navigation problem, and especially combinations of the last two methods in the list 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.

The focus here will be on the design and implementation of Kalman filters for these applications, including design 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.

There is yet another level of navigation accuracies required for applications such as surveying. These also use Kalman filtering with much the same approach, but with far more detailed modeling.

10.2.2 History of Inertial and Satellite Navigation

Inertial navigation was developed in the mid-twentieth century for missile guidance and matured during the Cold War as a technology for long-range delivery of nuclear weapons. Initial results were aimed at military applications such as self-contained guidance and control systems for ballistic and cruise missiles, submarines carrying ballistic missiles, military ships, and military aircraft, but the technology soon spread to nonmilitary applications such as commercial aircraft navigation.

Because of its stealth and immunity to countermeasures, inertial navigation is very well suited for ballistic missile guidance. Its time of operation during a missile launch is in the order of several minutes, which does not allow much time for any significant build-up of navigation errors. This has helped in achieving acceptable targeting accuracies using inertial navigation technology.

Submarines had long used gyroscopes to keep track of orientation while submerged, and integrating these with acceleration sensor was a natural progression to self-contained stealthy navigation. However, nuclear-missile-carrying submarines would require submerged operation for months, and this was too long to maintain inertial navigation accuracies sufficient for launching ballistic missiles. Some sort of auxiliary navigation information would be necessary for maintaining sufficient navigation accuracies.

Technology for satellite navigation developed soon after the 1957 launch of the world's first artificial satellite. The US Navy was quick to recognize that this could solve the long-range submarine navigation problem. The world's first satellite navigation system (Transit) would be developed and fielded for that specific purpose. After about a quarter century of military-only operation, it would be succeeded by a GPS satellite navigation system with limited civilian access to one of its signals. The impact on civilian navigation worldwide would be revolutionary, and many other countries would develop their own GNSS and augmentations thereof.

Although military satellite navigation systems were designed from the start for integration with INS, commercial development of civilian integrated GNSS/INS was much slower to develop—in part because the additional cost for inertial navigation[1] may not have been justified for the anticipated gain in performance. Among the first nonmilitary applications were such high payoff applications as automated surface mining and grading, but the lowered cost of inertial technologies sufficient for GNSS integration has expanded the number of potential applications.

10.2.3 GNSS Navigation

Satellite signal processing for GNSS navigation produces "pseudoranges" (imputed distances) between the receiver antenna and those of the satellites in view, the locations of which are known at all times. Using these pseudoranges as measurements, the Kalman filter generates estimates of the position and velocity of the GNSS receiver antenna with respect to Earth-fixed coordinates. To do so, it requires stochastic error models for the receiver clock, GNSS pseudorange data, and host vehicle dynamics. The "host vehicle" in this case could be a spacecraft, aircraft, watercraft, wheeled vehicle, or a pack animal (including humans). Each of these has different dynamic statistics, and the respective statistical characteristics can be exploited to improve navigation performance.

10.2.4 Integrated GNSS/INS Navigation

For navigation with an INS and other aiding sensors (e.g., altimeters, airborne radars, star trackers, or GNSS), the INS is the essential keeper of short-term navigation information and the Kalman filter functions to keep that navigation information as accurate as the noise sources will allow.

In the case of GNSS/INS integration, the filter replaces the stochastic model for unpredictable host vehicle dynamics with one for the INS and uses that model to

[1] In the 1970s, when airlines were first required to carry two inertial navigators for over-water flights, the increased equipment cost was in the order of $100,000 per airplane.

estimate, correct, and compensate for errors in the INS implementation. The Kalman filter inputs from the INS its navigation solution (position, velocity, and attitude) plus additional variables required for implementing the INS error model (acceleration and attitude rate), and outputs to the INS updates for its navigation solution (position, velocity, and attitude) plus updates to the parameters used for compensating inertial sensor errors. The result is a profound improvement in INS performance over the long term, as well as the usual short-term performance expected from inertial navigation. A key feature of the integrated navigator is its ability to maintain short-term accuracy when GNSS signals are not available.

10.2.5 Measures of Navigation Performance

10.2.5.1 The Nautical Mile (NMi) Historically, navigators at sea have used the *nautical mile* as the unit of choice when talking distances, and that notation has been passed along to landlubbers, as well. It was originally defined as the distance at sea level equivalent to one arc-minute of latitude change. This made sense as a unit of navigational uncertainty at a time when Earth was thought to be spherical, latitude was determined by measuring the angle of the Pole Star above the horizon at sea, and a minute of arc was the approximate limiting resolution of optical sighting instruments at sea.

However (as Newton surmised) Earth is not quite spherical in shape. As a consequence, the north–south distance equivalent to a variation of one arc-minute of latitude varies by several meters. As a fix, *Le Système international d'unités* (SI) defined the nautical mile as a derived SI unit, equivalent to 1852 m. This would be about 1.15078 US statute miles or 6076.12 US feet.

10.2.5.2 Circle of Equal Probability (CEP) It is also called *circular error probable* (CEP), defined as the radius of a circle centered at an estimated location on the surface of Earth such that it is equally likely that the true location is either inside or outside to that circle.

It is a useful notion in navigation and targeting, giving a precise meaning to a single number characterizing accuracy.

Its implementation in Kalman filtering can be troublesome, however, because one cannot translate estimated position error covariances into CEP without assuming something more about the probability distribution than its mean and covariance. The problem is commonly handled by assuming the probability distributions are Gaussian and approximating CEP as 1.2 times the root-mean-square (RMS) radial horizontal error.

10.2.6 Performance Prediction in Navigation System Design

10.2.6.1 Covariance Analysis The Kalman filter serves a dual role in estimation technology, in that it is not only an optimal estimator, but also serves to predict performance based on the dynamics of the variables being estimated and the statistical characteristics of the sensor system to be used for estimation.

The same Kalman filter models developed for navigation implementation may also be used to design a navigation system to be used on an ensemble of trajectories of the host vehicle carrying the sensors. The same general approach applies to the design of navigation satellite systems, inertial navigators, and integrated GNSS/INS navigation systems.

In the first case (GNSS), the Kalman filter not only functions as a design evaluation tool but also plays an essential role in system integration.

In the second case (INS), parametric INS models are used to represent the possible choices of inertial sensors in the INS, and representative trajectory simulations can be used for comparing relative performances of the design alternatives.

In the case of GNSS/INS integration, the Kalman filter model can be used to assess the performance trade-offs for alternative INS sensor capabilities. In addition, the design problem for integrated GNSS/INS navigation may include ancillary performance objectives such as

1. Minimizing the cost of the onboard integrated GNSS/INS system.
2. Achieving specified "stand-alone" INS-only performance when GNSS signals are lost.
3. Reduction of GNSS signal reacquisition time.
4. Using INS information to improve GNSS signal lock, especially during severe dynamic maneuvers of the host vehicle or periods with excessive satellite signal interference.

10.2.6.2 Test and Evaluation The covariance analysis used for predictive design must eventually be verified by testing and evaluation under controlled conditions. For systems designed for specific applications, the test conditions generally match the intended applications. There are also formal test and evaluation procedures describing what testing is done, what data is taken, how the data is processed, and what specific performance metrics are to be evaluated. For military aircraft navigation systems, this has been done for more than half a century at the Central Inertial and GPS Test Facility (CIGTF) at Holloman Air Force Base near Alamogordo, New Mexico. The facility includes laboratory equipment for evaluating systems under controlled environmental conditions, precision tracking systems for systems under flight or ground testing, formal test and evaluation procedures for specific military applications, and data processing methods approved for evaluating performance relative to the established military standards.

10.2.6.3 Additional Information Sources For a broader historical background on the subject, see, for example, Reference 1. For broader technical coverage of the subject, see, for example, Reference 2. For more in-depth coverage of inertial navigation, see, for example, References 3–5. For better coverage of GNSS receiver technology, see, for example, Reference 6.

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. Searches for "CIGTF" may yield more formal descriptions of test and evaluation procedures for INS, GNSS, and integrated GNSS/INS navigation systems.

10.2.7 Dynamic Simulation for Predicting Navigation Performance

Prior to any laboratory or field testing, a common set of simulated dynamic conditions can be used for performance prediction of GNSS navigation, INS navigation, and integrated GNSS/INS navigation. Performance of any developing navigation system can be assessed by testing and evaluation under the intended operating conditions, including dynamic conditions. Assessing expected performance requires some sort of dynamic simulator to generate the essential inputs to the navigation solution under those conditions. As an example, MATLAB® simulators are derived, encoded in m-files on the Wiley web site, and used in demonstrating navigation performance for GNSS, INS, and integrated GNSS/INS navigators.

10.2.7.1 Stationary Testing Checkout of a navigation system usually begins by running it in a known fixed location. For GNSS navigation, this must include a simulator for the satellite locations over time. Even inertial navigation, which is essentially differential, is usually verified first in the laboratory, before dynamic testing. In all cases, the Kalman filter for this is relatively simple and is used to demonstrate the effects of different error sources under "neat" conditions, without the corrupting effects of host vehicle dynamics. For an INS, this sort of testing is used for verifying gyrocompass alignment and for verifying that the INS and its error model show the same behaviors when initial errors are introduced.

10.2.7.2 Racetrack Simulators The common host vehicle dynamic model used for most simulations is for a host vehicle running at 100 kph average speed on a figure-8 track of various lengths. Two track layouts are illustrated in Figure 10.2 for two specific track lengths, although some versions of the track simulator allow both the track length and vehicle speed to be specified at runtime. The limited position excursions of the track are exploited in designing the stochastic dynamic model for the host vehicle used in the associated Kalman filter for by solving the navigation problem.

A vehicle running at 100 kph hardly qualifies as a "racecar," but integrated GNSS/INS navigation is not uncommon on racecars 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 racecar is at all times. 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 in the order of a meter, which is sufficient for this purpose. The heading and pitch angles of the optical axes of track television cameras are used together with camera lens focal lengths to compute where each racecar 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

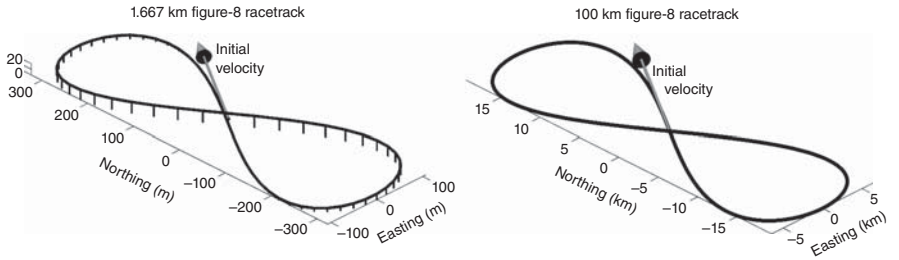


Figure 10.2 Figure-8 track layouts.

track location model uses only along-track position. The m-file `GNSSshootoutNCE.m` demonstrates how this level of modeling can achieve RMS position uncertainties that are 30–50 times smaller than those from using more conventional host vehicle dynamic models.

Statistics of Racetrack Dynamics Average vehicle dynamic statistics during figure-8 track simulations have the values shown in Table 10.1 for the track layouts shown in Figure 10.2. To provide empirical values of dynamic disturbance covariance Q in the Kalman filter models for GNSS navigation, the RMS velocity changes were sampled at the GNSS navigation update intervals on track simulations.

These statistics are matched to the parameters of the stochastic dynamic models used for representing host vehicle dynamics during GNSS navigation.

10.2.7.3 Simulating GNSS Signal Loss GNSS signal availability depends on having an unobstructed line of sight from the transmitting satellite to the receiver antenna and no signal interference at the receiver antenna from sources such as unintended (or intended) interference or multipath reflections. These conditions can fail for a number of reasons. In most cases, signal interruptions are only temporary, however. Still, when GNSS signals are lost, the navigation solution can deteriorate rapidly, with the rate of deterioration generally depending on the relative predictability of host vehicle dynamics. Simulated test conditions for evaluating performance degradation after loss of signal used a modified figure-8 test track simulator with a tunnel covering just a portion of the vehicle trajectory, as illustrated in Figure 10.3. The simulator assumes 100-kph host vehicle speed around a 100-km figure-8 track, which equates to a lap time of 1 h. The tunnel is assumed to cover the track for 1 min of travel, ending 1 min before the undercrossing.

10.3 GLOBAL NAVIGATION SATELLITE SYSTEMS (GNSS)

10.3.1 Historical Background

As we have mentioned in the previous section and in Chapter 1, the beginnings of practical satellite navigation technology occurred soon after America's original

TABLE 10.1 Dynamic Statistics on Figure-8 Track Simulators

	FIG8 Track*	BIG8 Track	
Statistic	Value	Value	Units
Track length	1.667	100.0	km
Average Speed	100.0	100.0	kph
RMS N–S Position Excursion	230.2564	13,699.7706	m
RMS E–W Position Excursion	76.7521	4,566.5902	m
RMS Vertical. Position Excursion	12.3508	6.1237	m
RMS N–S Velocity	24.3125	23.9139	m/s
RMS E–W Velocity	16.2083	15.9426	m/s
RMS Vertical Velocity	0.74673	0.0061707	m/s
RMS N–S Acceleration	2.525	0.041732	m/s/s
RMS E–W Acceleration	3.3667	0.055643	m/s/s
RMS Vertical Acceleration	0.078846	1.0771_{10}-5	m/s/s
RMS Delta Velocity North†	2.5133	0.041732	m/s/s
RMS Delta Velocity East†	3.3464	0.055642	m/s/s
RMS Delta Velocity Up†	0.077832	1.077_{10}-5	m/s/s
RMS Roll Rate	0.0092193	‡	rad/s
RMS Pitch Rate	0.06351	‡	rad/s
RMS Yaw Rate	0.15311	‡	rad/s
RMS Delta Roll Rate†	0.0022114	‡	rad/s/s
RMS Delta Pitch Rate†	0.010502	‡	rad/s/s
RMS Delta Yaw Rate†	0.023113	‡	rad/s/s
North Position Correlation Time	13.4097	735.6834	s
East Position Correlation Time	7.6696	354.155	s
Vertical Position Correlation Time	9.6786	1,661.1583	s
North Velocity Correlation Time	9.6786	635.1474	s
East Velocity Correlation Time	21.4921	354.155	s
Vertical Velocity Correlation Time	13.4097	735.6834	s
North Acceleration Correlation Time	13.4097	735.6834	s
East Acceleration Correlation Time	7.6696	354.155	s
Vertical Acceleration Correlation Time	9.6786	635.1474	s

∗ Values shown are for strapdown version `Fig8TrackSimRPY.m`.
Track length and speed can be varied through input variables of other versions.
† Sampled at 1-s intervals.
‡ Not computed for gimbaled INS simulation.

"Sputnik moment," the October 4, 1957 launch of the world's first artificial satellite[2] by the former Soviet Union. This soon lead to the development by the US Navy of the Transit satellite navigation system, the primary function of which would be as an aid in improving inertial navigation capabilities of military systems—especially for nuclear-missile-carrying submarines. However, due to demands of military secrecy and security, its capabilities would not be made available to the general public. This restriction was eased somewhat in the 1980s for Transit's eventual replacement, the

[2]"Sputnik" is a transliteration of the Russian word for satellite.

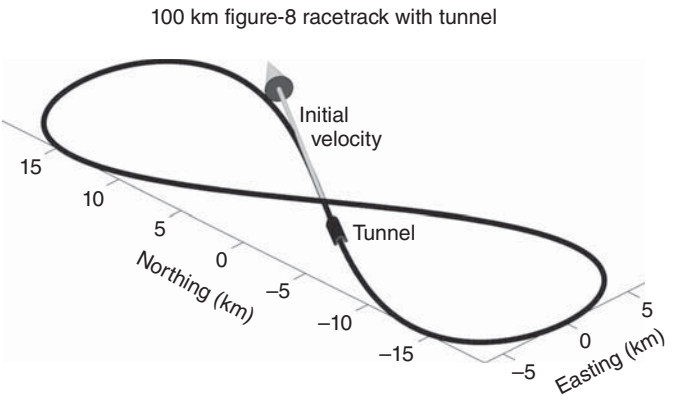


Figure 10.3 Simulator for loss of signal.

Global Positioning System (GPS). On September 1, 1983, inertial navigation errors aboard Korean Air Lines Flight 007 caused it to pass briefly through Soviet airspace, which ended in its being destroyed in international airspace by military aircraft of the former Soviet Union. After that, US President Ronald Reagan signed a directive requiring that the then-developing GPS satellite navigation system be accessible by the general public. The result would revolutionize general navigation and lead to similar efforts by other countries and agencies. To date (2014), the only completely independent functioning alternative to GPS is the Russian GLONASS[3] satellite navigation system, but the European Union (EU) and European Space Agency (ESA) are planning to reach operational status with their own system (Galileo) in 2019, and China plans on expanding its developing regional navigation system (Beidou) into a global system (Compass) by 2020. In addition, France, Japan, and India are developing their own regional augmentation systems.

These systems generally contain ground-based assets (ground segment) integrated with space-based assets (space segment) using Kalman filtering. These Kalman filters are an integral part of the GNSS software infrastructure, but there is also a need for independent Kalman filters in the associated GNSS receivers, for estimating the navigation solution (including the receiver antenna location) based on the satellite "signals in space."

This section is about receiver-based Kalman filter architectures for obtaining the GNSS navigation solution. There is far more diversity in Kalman filter designs at the receiver level, because performance at that level depends on specifics of the intended application(s).

10.3.2 How Satellite Navigation Works

10.3.2.1 Doppler-based Solution As described in Chapter 1, the Transit navigation satellite system used as its basic measurement variables the Doppler shifts in

[3]Except for an 11-h outage in April Fool's Day of 2014.

satellite signals as the satellites passed overhead. Its navigation solution method used linearized least-squares fitting (later Kalman filtering) of the antenna position to an observed sequence of Doppler shifts from a satellite with known trajectory. Transit was originally designed for military applications in which the receiver antenna was essentially stationary during the few minutes required for a satellite to pass from horizon to horizon. In the 1970s and 1980s, Transit and Timation (another US Navy satellite system) also served as a testbeds in the development of extremely accurate ground- and space-based clock technologies[4] for the next generation of satellite navigation systems. That would be the US Air Force's GPS, which became operational around the time Transit was ready for retirement in the 1990s. All satellite navigation systems developed thereafter are based on accurate timing technologies.

10.3.2.2 Timing-Based Solution After sufficient timing capabilities had been developed, satellite navigation solutions became "timing based." That is, they use a system of synchronized clocks to measure the time it takes an electromagnetic signal to pass from the transmitting antenna on a satellite to the receiving antenna. Given the transit time Δt and the speed of electromagnetic wave propagation c, this gives a measure of the distance $\rho = c\ \Delta t$ between the two antennas. If the locations of three transmitting antennas on three satellites are known as *functions of time*, this would allow a solution for the location of the receiver antenna, as illustrated in Figure 10.4. That is, given the location in space of the three satellites forming the triangular base of the tetrahedron formed by the four antennas and the three legs ρ_i, there is a unique solution for the location of the receiver antenna. In practice, however, more than three satellites are needed for maintaining clock synchronization.

10.3.3 GNSS Error Sources

A Kalman filter needs error models for all the measurements it uses. For GNSS receivers, the basic measurements are propagation delay times used for computing the distances between the satellite antennas and the receiver antenna. The associated measurement error models have been determined from theoretical and empirical analysis of the GNSS system as it developed.

10.3.3.1 Space Segment Errors A satellite navigation system is infected with all sorts of error sources that have been partially cured by Kalman filtering , a side benefit of which is that the resulting RMS errors of the various contributors to navigation

[4]This was not the first use of accurate timing for navigation. The need for accurate timing to determine longitude at sea was well known as far back as the seventeenth century, when the pendulum clocks of the day would not function at sea. England was then losing more of its naval ships from grounding than from combat, and the disastrous grounding of four Navy ships on the Isles of Scilly in 1704 only emphasized the importance of a speedy solution. After the formation of the United Kingdom in 1707, its Parliament passed the Longitude Act of 1714, establishing substantial prizes to be awarded by Commissioners for the Discovery of the Longitude at Sea (1714–1828) for specific levels of timing accuracy and reliability. The problem was eventually solved by the world's first chronometers, designed by John Harrison (1693–1776). However, awarding of any prizes was thwarted by Nevil Maskeline (1732–1811) and the Commissioners until King George III (1738–1820) interceded on Harrison's behalf [7]

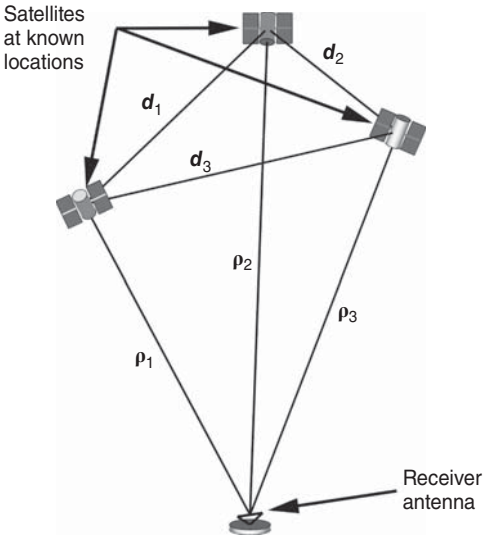


Figure 10.4 Minimal timing-based satellite navigation solution geometry.

errors can be determined from covariance analysis, using the associated solutions of the Riccati equation. For GPS, for example, location errors due to the precise ephemerides (trajectories) of individual satellites (broadcast by all satellites) is in the order of 2 m RMS, as is the error contributed by synchronization of all satellite-based clocks. The current GLONASS K-class space segment performance is comparable.

10.3.3.2 Propagation Delay Errors From the GNSS user's standpoint, the biggest error sources are due to variations in signal propagation delays between the satellite antennas and the receiver's antenna. These are dominated by the effects of free electron density in the ionosphere, which varies considerably with solar radiation and space weather. The effect of this variation on the apparent transmission distance is in the order of 10 m RMS—but this can vary up and down by an order of magnitude, depending on levels of solar activity [8]. The dynamics of propagation delay errors at the receiver are also driven by the motions of the satellites in orbit, which drags the path of the signal from satellite to receiver through different parts of the ionosphere over time.

GPS receivers use different methods for mitigating the effects of these errors:

1. Global correction formulas based on tomographic models of the atmosphere, and using measurements made at ground stations at many locations. These stations know where they are, so they can estimate the ionospheric delay along the paths between the station antennas and satellite antennas. There are several such formulas used in existing or proposed augmentation systems associated with different GNSS systems. A formula due to Klobuchar [9] is included in most single-frequency GPS receivers, and it uses parameters broadcast as part

of the GPS signal. This can reduce the contribution to the user location error by a factor of about 2 or 3, but even this can be improved upon by augmenting the state vector of the Kalman filter in the receiver used for estimating the navigation solution. The augmenting state variables in this case will be the residual errors from the Klobuchar model corrections, which are time-correlated with effective correlation times are in the order of minutes to hours [10].

2. A dual-frequency correction method is based on the known relationship between primary signal delay and differential delay between the two different gigahertz carrier frequencies separated by several hundred megahertz. This feature was originally included in GPS as a military-only option, which required a decryption key to access the timing information on the second frequency. However, current GPS upgrades and other GNSS[5] systems include unencrypted civilian signal frequencies (L2C and L5 for GPS) that serve the same purpose. This approach reduces the ionospheric delay contribution to something in the order of 1 m RMS, which is small enough that the residual ionospheric propagation delay errors may not need to be estimated using the receiver's Kalman filter.
3. Differential GNSS navigation using propagation delay values calculated at local ground-fixed receiver stations with known locations and transmitted on a separate channel for use by local GNSS receivers. The US Coast Guard inaugurated such a service for GPS around major waterways and ports in the United States in the 1990s. This nulled out the "Selective Availability" measures the US Department of Defense had designed into the GPS civilian channels to derate RMS navigation performance to around 100 m RMS. Similar services were designed into various Wide Area Augmentation Systems (WAAS) by the United States and other countries. Receivers equipped for differential GPS navigation could then achieve RMS horizontal navigation accuracies of 5 m or less. Selective Availability was removed from GPS in 2000.

There is also a tropospheric propagation delay error and errors due to atmospheric refraction, but their contributions are generally less than 1 m RMS.

Propagation delays also depend on the atmospheric propagation path length, which depends on the slant distance through the atmosphere from the satellite to the receiver antenna. This can make signal propagation delay errors from satellites nearer to the horizon two to three times what they are for satellites directly overhead.

10.3.3.3 Receiver Clock Errors Clocks are among the most analyzed and improved devices we make and use every day, and their behavior over time is fairly well understood and modeled. GNSS receivers can get by with relatively inexpensive quartz resonator clocks, using timing from satellite- and ground-based "atomic clocks" to maintain GNSS system-wide synchronization. This approach

[5]GLONASS is in the process of changing over from a frequency-division multiple access (FDMA) protocol to a code-division multiple access (CDMA) protocol with a common carrier frequency and different spreading codes for different satellites. GPS had been designed from the start to use CDMA with "Gold codes" (designed by Robert Gold) to minimize cross-interference of the spread-spectrum signals.

takes advantage of the superior short-time stability of quartz clocks. An essential part of this is a linear stochastic process model for a receiver clock phase and frequency error and their respective uncertainties as functions of time. The problem is solved by making the clock *bias* (phase error) and *drift* (frequency error) part of the navigation solution.

10.3.3.4 Dilution of Precision (DOP) The minimal essential state variables for a single GNSS location solution are three components of receiver antenna position, plus receiver clock error. An estimate of clock error is essential because it is too big to be ignored. Other "nuisance variables" may be estimated to accommodate other time-correlated error sources, but these four are the minimal set necessary for obtaining a single location solution. How well these four variables can be estimated will be determined by timing measurement noise and satellite geometry. The term *dilution of precision* (DOP) was defined to characterize how relative beacon locations influence LORAN navigation accuracy, and it has been extended to characterize how satellite location geometry affects GNSS navigation accuracy. In both cases, it can be defined in terms of the related measurement information matrices.

GNSS Timing Measurement Sensitivities and Information Matrices As described in Example 8.7, the measurement sensitivity matrix for the timing variation with respect to receiver position is a unit vector u_j in the direction from the receiver antenna to the jth satellite. If clock error is defined as phase lag and multiplied by c to make it an equivalent distance, then the measurement sensitivity matrix for position and clock error for a single timing measurement is

$$H_j = \begin{bmatrix} u_j^{\mathrm{T}} & -1 \end{bmatrix}, \tag{10.1}$$

the measurement sensitivity matrix for N such measurements is

$$H = \begin{bmatrix} u_1^{\mathrm{T}} & -1 \\ u_2^{\mathrm{T}} & -1 \\ u_3^{\mathrm{T}} & -1 \\ \vdots & \vdots \\ u_N^{\mathrm{T}} & -1 \end{bmatrix}, \tag{10.2}$$

and the associated information matrix for the N measurements is

$$Y = H^{\mathrm{T}} R^{-1} H \tag{10.3}$$

$$= R^{-1} \sum_{j=1}^{N} \begin{bmatrix} u_j \\ -1 \end{bmatrix} \begin{bmatrix} u_j \\ -1 \end{bmatrix}^{\mathrm{T}}, \tag{10.4}$$

where the scalar parameter R is the mean-squared measurement error.

Position and Clock Error Estimation Covariance The resulting covariance matrix of state estimation uncertainty will be the inverse of the associated information matrix

$$P = Y^{-1} \tag{10.5}$$

$$= R\mathcal{D} \tag{10.6}$$

$$\mathcal{D} = \left(\sum_{j=1}^{N} \begin{bmatrix} u_j \\ -1 \end{bmatrix} \begin{bmatrix} u_j \\ -1 \end{bmatrix}^{\mathrm{T}} \right)^{-1} \tag{10.7}$$

$$= \begin{bmatrix} d_{11} & d_{12} & d_{13} & d_{14} \\ d_{21} & d_{22} & d_{23} & d_{24} \\ d_{31} & d_{32} & d_{33} & d_{34} \\ d_{41} & d_{42} & d_{43} & d_{44} \end{bmatrix}. \tag{10.8}$$

The 4×4 matrix $\mathcal{D}$ represents the multiplying factor between mean-squared measurement noise R and mean-squared uncertainty of estimated location and clock error due to satellite geometry.

The term *dilution of precision* (DOP) refers to the related multiplying effect of RMS (as opposed to mean-squared) measurement errors on RMS estimation errors due to satellite geometry. The overall effect is called *geometric dilution of precision*, and it is characterized by the square root of the matrix trace of $\mathcal{D}$.

Better yet, if the unit vectors u_j to the respective satellites used in the estimation are expressed in east-north-up coordinates, the square roots of the successive diagonal elements of $\mathcal{D}$ characterize the DOP with respect to east position, north position, up position (vertical), and clock error uncertainties. This convention leads to various "sub-DOPs" defined in terms of the respective diagonal elements of $\mathcal{D}$ as

$$\left.\begin{array}{lcll} \mathrm{GDOP} & \stackrel{\mathrm{def}}{=} & \sqrt{\mathrm{tr}\,(\mathcal{D})} & \text{(Geometric DOP)} \\ & \stackrel{\mathrm{def}}{=} & \sqrt{d_{11} + d_{22} + d_{33} + d_{44}} & \\ \mathrm{PDOP} & \stackrel{\mathrm{def}}{=} & \sqrt{d_{11} + d_{22} + d_{33}} & \text{(Position DOP)} \\ \mathrm{HDOP} & \stackrel{\mathrm{def}}{=} & \sqrt{d_{11} + d_{22}} & \text{(Horizontal DOP)} \\ \mathrm{VDOP} & \stackrel{\mathrm{def}}{=} & \sqrt{d_{33}} & \text{(Vertical DOP)} \\ \mathrm{TDOP} & \stackrel{\mathrm{def}}{=} & \sqrt{d_{44}} & \text{(Time DOP)} \end{array}\right\} \tag{10.9}$$

10.3.3.5 Multipath Effects "Multipath" is the name used for GNSS signal distortion from signal reflections off surfaces such as the ground or sea surface. Antenna gain patterns of many GNSS receivers are designed to attenuate signals on or near the horizon, but this will not correct for reflections off mountains, hills, buildings, or vehicle parts protruding above the antenna. This problem can be especially severe in "urban canyons" lined with flat-faced buildings, and several signal processing methods have been developed for contending with the problem [2].

It is not usually treated as a Kalman filtering problem, but as a problem to be solved in the signal processing "upstream" of the Kalman filter used for obtaining the navigation solution.

10.3.3.6 Other Receiver Errors

Receiver Noise Most GNSS receivers have been designed so that all the error contributions from receiver signal processing generally amount to less than 1-m RMS position error.

10.3.4 GNSS Navigation Error Modeling

Navigation errors not tamed by the Kalman filters in the ground segment or space segment of a GNSS system must be tamed by a Kalman filter in the receiver. The Kalman filters in the ground and space segments have already been designed and implemented before a GNSS becomes operational. This section is about those receiver-based models used for navigation and also used for integrating GNSS receivers with other navigation sensors.

10.3.4.1 GNSS Navigation Measurement Model

Pseudoranges The tetrahedral solution shown in Figure 10.4 would work just fine if all the necessary information were error-free—which it is not. Even if the locations of the satellites were know precisely (which is almost true), the ranges from the receiver antenna to satellite antennas determined by timing are called *pseudoranges*, because they do contain errors. The major error sources requiring correction can be modeled in terms of their effects on pseudorange errors.

The real situation is more like that shown in Figure 10.5, where all available satellites above the horizon (shown connected to the receiver by solid lines) are used in the solution, and the actual distances to the respective satellites are more like

$$\rho_j = c \times (t_{j,\text{received}} - t_{j,\text{transmitted}} - \delta t_{j,\text{iono delay modeled}} - \delta t_{j,\text{iono delay unmmodeled}} - \delta t_{\text{receiver clock bias}}), \tag{10.10}$$

where the variables are defined as

- ρ_j, the physical distance between the antenna on the jth satellite at time $t_{j,\text{transmitted}}$ and the antenna on the GNSS receiver at time $t_{j,\text{received}}$.
- $c \overset{\text{def}}{=} 299,792,458$, the speed of light in SI units.
- $t_{j,\text{received}}$, the time at which the timing mark on the jth satellite signal was received at the receiver antenna.
- $t_{j,\text{transmitted}}$, the time at which the timing mark on the jth satellite signal was transmitted from the jth satellite antenna.
- $\delta t_{j,\text{iono delay modeled}}$, the modeled mean propagation delay over the interval $[t_{j,\text{transmitted}}, t_{j,\text{received}}]$, based on

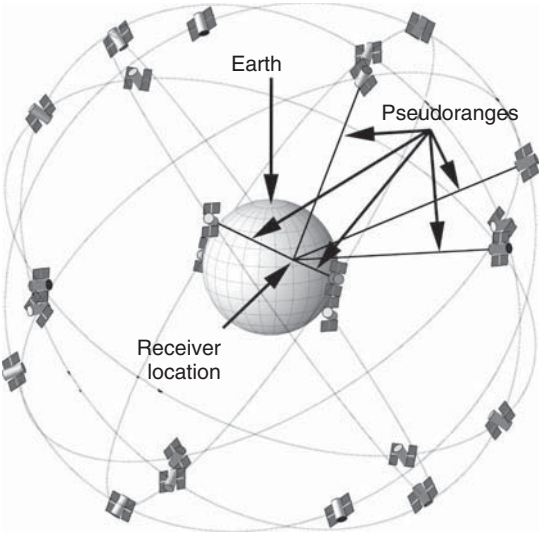


Figure 10.5 Satellite navigation geometry.

(a) a Klobuchar parametric model with parameter values transmitted by the satellites. These parameters are determined in real time from timing measurements made by many receiver ground stations covering much of the land surface, and the resulting model removes most of the ionospheric delay (but not enough),

(b) a formula using signals from the same satellite at two different frequencies, or

(c) differential delay corrections determined at a local auxiliary GNSS receiver at a known fixed location, and broadcast to local GNSS receivers.

$\delta t_{j,\text{iono delay unmmodeled}}$, the unmodeled propagation delay due to limitations of a parametric Klobuchar-like model. This error term may be ignored for the dual-frequency and differential approaches, which may correct enough of the delay that secondary filtering is not necessary.

$\delta t_{\text{receiver clock bias}}$, the bias in the receiver clock at the time the signal was received.

In practice, the last two of these (unmodeled ionospheric delay and receiver clock bias) can be estimated in the same Kalman filter used for estimating the receiver antenna location, and the scaling by c is used to convert all timing error variables to equivalent distance variables. All the other variables are already known from other sources.

This leaves some unknown random variables that require stochastic dynamic models for their time correlation and will add to the number of state variables to be estimated in the Kalman filter.

10.3.4.2 Measurement Sensitivity Matrix Structure The GNSS measurement is a pseudorange with added error variables that corrupt the measurement, but these can also be estimated and compensated for. These error variables include clock errors, and possibly uncompensated ionospheric delay errors (depending on what mitigation is used).

Distributing the factor of c over the right-hand side of Equation 10.10 yields a measurement function in distance units:

$$\rho_j = \underbrace{c\,(t_{j,\text{received}} - t_{j,\text{transmitted}} - c\,\delta t_{j,\text{iono delay modeled}})}_{\rho_{0,j}} \underbrace{-c\,\delta t_{j,\text{iono delay unmmodeled}}}_{\delta_{\text{iono},j}} \underbrace{- c\,\delta t_{\text{receiver clock bias}}}_{C_b}, \tag{10.11}$$

where C_b is the error in the pseudorange measurement attributable to clock bias, $\delta_{\text{iono},j}$ is the error attributable to uncompensated ionospheric propagation delay, and $\rho_{0,j}$ is the (relatively) error-free measurement component—attributable to the physical distance between the satellite antenna and the receiver antenna.

The pseudorange error $\delta_{\text{iono},j}$ due to uncompensated ionospheric delay is a "nuisance variable," in that we need it to improve navigation accuracy, but otherwise do not care what its value is.

Clock bias C_b is a little different, in that it is a variable used in compensating all timing calculations. It is also part of a model used for compensating clock frequency, which is also an essential part of the receiver implementation.

The associated measurement sensitivity to the five state variables (three antenna position components, clock bias, and uncompensated ionospheric delay error) is then

$$H_j \overset{\text{def}}{=} \frac{\partial \rho_j}{\partial x_{\text{antenna}}, \ldots, C_b, \ldots, \delta_{\text{iono},j}} \tag{10.12}$$

$$= \begin{bmatrix} u_j^{\mathrm{T}} & \cdots & -1 & \cdots & -1 & \cdots \end{bmatrix}, \tag{10.13}$$

where the unspecified elements of the row vector H_j are zeros and

x_{antenna} is the location of the receiver antenna.

u_j is a unit vector pointing to the satellite from the receiver antenna.

the first "-1" represents the sensitivity to receiver clock bias.

the second "-1" represents the sensitivity to uncompensated propagation delay for the jth satellite.

10.3.4.3 Dynamic Model Structure These models are all defined by the resulting values of the matrices F (or Φ), Q, H, and R to be used in the Kalman filter.

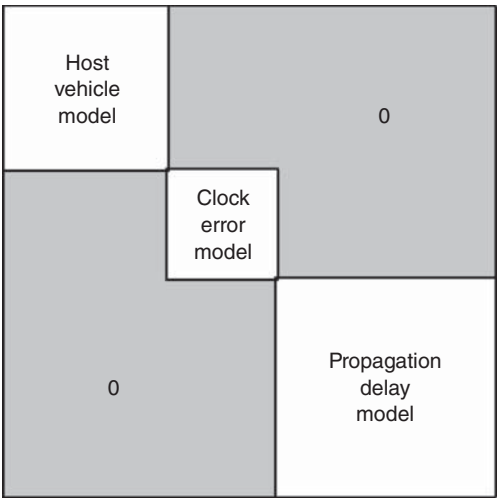


Figure 10.6 Structure of Kalman filter model F-matrix and Q-matrix for GNSS navigation.

The structure of the F and Q matrix will be as shown in Figure 10.6. This block-diagonal structure is useful in practice, because

$$\Phi_{k-1} = \exp\left[\int_{t_{k-1}}^{t_k} F(s)\, dx\right] \tag{10.14}$$

will have the same block diagonal structure as F, with the diagonal subblocks equal to the matrix exponentials of the relevant subblocks of F. This allows the filter designer to treat the dynamic submodels independently, taking exponentials of the time-invariant submatrices only once.

10.3.4.4 Exponentials of Block-Diagonal Matrices There are useful ways to reduce the computational effort required for taking exponentials of matrices with block-diagonal structure such as that shown in Figure 10.6—especially when only a proper subset of the submatrices are time varying. For any matrix with block-diagonal structure

$$M = \begin{bmatrix} M_{11} & 0 & \cdots & 0 \\ 0 & M_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & M_{NN} \end{bmatrix}, \tag{10.15}$$

$$\exp\,(M) = \begin{bmatrix} \exp\,\left(M_{11}\right) & 0 & \cdots & 0 \\ 0 & \exp\,(M_{22}) & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & \exp\,(M_{NN}) \end{bmatrix}. \tag{10.16}$$

The number of scalar arithmetic operations required for computation of matrix exponentials grows as the cube of the matrix dimension, so partitioning the matrix in this way saves considerable time and effort.

If the matrix M in this case is the dynamic coefficient matrix F for a linear dynamic system, the associated state-transition matrix is defined by Equation 10.14. Consequently, if any diagonal block F_{ii} of F is time invariant, its exponential

$$\Phi_{(k-1)\ ii} = \exp\ \left(\int_{t_{k-1}}^{t_k} F_{ii}\ dt \right) \tag{10.17}$$

$$= \exp\ (F_{ii}\ \Delta t) \tag{10.18}$$

is also time invariant.

This will be the case for all diagonal submatrices in the dynamic coefficient matrices used for solving the GNSS navigation problem. However, it will not be the case when we get to integrating GNSS with INS. In that case, only the time-varying blocks associated with inertial navigation error propagation need to be recomputed as a function of time. The blocks with time invariant F_{ii} can be left unmolested.

Dimensions of these diagonal subblocks will be determined by which of several possible models is chosen, as will the dimensions of the H and R matrices.

10.3.4.5 Host Vehicle Dynamic Models Timing-based GNSS navigation is defined in terms of having a certain receiver antenna location at the time a satellite signal timing mark is received, and the timing-based navigation correction component for that measurement is always in the direction between the satellite and the receiver location. If the antenna were to remain in that same location long enough for similar corrections to be make for all available satellites, then (assuming good GDOP) that location would be observable from the ensemble of measurements. This may not be a problem in surveying applications, for which the receiver antenna is held at a fixed position on Earth. It is a problem for other applications, however.

The GNSS geometry shown in Figure 10.5 takes into account the motion of the satellite antennas, which is determined by the short-term ephemerides transmitted by the satellites. There is no equivalent predetermined trajectory for the receiver antenna.

Tracking Filters The problem gets a bit more complicated for navigation aboard maneuvering vehicles, in that the location of the receiver antenna now becomes an unknown function of time. This is the same sort of problem faced in the early 1950s, when radar systems were being integrated with real-time computers for detecting and tracking aircraft as part of an air defense system [11], and the GNSS navigation solution can use the same types of "tracking filters" developed around that time. The Kalman filter was not available in the 1950s, but today's tracking filters use both the position and velocity of the receiver antenna[6] as state variables in a Kalman

[6]Air defense jargon uses the term *target* for the object being tracked.

filter. In the case of GNSS navigation, performance can be improved if the parameters of the filter are well matched to the dynamics of the host vehicle carrying the receiver antenna. These parameters include the dynamic coefficient matrix F and the mean-squared dynamic disturbance covariance Q.

Mathematical Formulas Table 10.2 lists some of the parametric models used for modeling a single component of the motion of a host vehicle. These are specified in terms of the dynamic coefficient matrix F and disturbance noise covariance Q for a single axis of 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.

Model Descriptions The model parameters listed in Table 10.2 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 Reference 2 for more details.)

TABLE 10.2 Host Vehicle Dynamic Models

Model No.	model parameters (each axis) F	Q	independent parameters	dependet parameters
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

The choice of a model and the values of its parameters will depend on the dynamic capabilities of the host vehicle and/or its likely trajectories in the navigation problem being considered.

1. *Stationary Model.* The first model in Table 10.2 is for an object fixed to the earth, such as a GNSS antenna at a fixed location. 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.
2. *Quasi-Stationary Model.* The second model in Table 10.2 is that 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 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 quasi-stationary 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 the 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^2_{acc} by the equations

$$\delta = \frac{2\,\sigma_{\text{vel}}}{\sigma_{\text{pos}}} \tag{10.19}$$

$$\sigma^2_{\text{acc}} = \frac{4\,\sigma^3_{\text{vel}}}{\sigma_{\text{pos}}}. \tag{10.20}$$

   The units of σ^2_{acc} 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.

   This is a good model for the adaptive suspension systems found in aircraft.

   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 medium accuracy strapdown inertial navigation systems.
3. *Type 2 Tracker.* The third model in Table 10.2 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 position and velocity of a host vehicle.

   The only adjustable parameter is the mean-squared acceleration noise σ^2_{acc}. 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.

4. *Modified Type 2 Tracker.* The fourth model in Table 10.2 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.

5. *Models 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.

6. *Models for Bounded RMS Position.* The last model in Table 10.2 is for vehicles with bounded position, as well. This model may be appropriate for the limited altitude excursions of ships or land vehicles, including vehicles operating within tightly confined areas.

Alternative Control-Based Model There is a related controls model for bounding the RMS position, velocity, and acceleration of a vehicle disturbed by white jerk noise:

$$\frac{d}{dt}\begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ f_{3,1} & f_{3,2} & f_{3,3} \end{bmatrix}\begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ w_{\text{jerk}}(t) \end{bmatrix} \tag{10.21}$$

$$f_{3,1} = -\frac{\sigma_{\text{jerk}}^2 \sigma_{\text{vel}}^2}{2(\sigma_{\text{pos}}^2 \sigma_{\text{acc}}^2 - \sigma_{\text{vel}}^4)} \tag{10.22}$$

$$f_{3,2} = -\frac{\sigma_{\text{acc}}^2}{\sigma_{\text{vel}}^2} \tag{10.23}$$

$$f_{3,3} = -\frac{\sigma_{\text{pos}}^2 \sigma_{\text{jerk}}^2}{2(\sigma_{\text{pos}}^2 \sigma_{\text{acc}}^2 - \sigma_{\text{vel}}^4)} \tag{10.24}$$

$$\sigma_{\text{jerk}}^2 = \text{E}_t\langle w_{\text{jerk}}^2(t)\rangle \tag{10.25}$$

$= \text{mean-squared jerk noise}$

$\sigma_{\text{pos}} = \text{RMS position excursion}$

$\sigma_{\text{vel}} = \text{RMS velocity}$

$$\sigma_{\text{acc}} = \text{RMS acceleration}$$
$$\sigma_{\text{pos}}^2 \sigma_{\text{acc}}^2 > \sigma_{\text{vel}}^4. \tag{10.26}$$

This can be verified by solving the associated steady-state Riccati equation,

$$0 = FP + PF^{\text{T}} + Q \tag{10.27}$$

$$Q = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \sigma_{\text{jerk}}^2 \end{bmatrix} \tag{10.28}$$

for P, giving the diagonal elements as the steady-state σ_{pos}^2, σ_{vel}^2 and σ_{acc}^2. The result can be solved for F as a function of σ_{pos}^2, σ_{vel}^2, σ_{acc}^2, and σ_{jerk}^2.

Empirical Modeling For most host vehicles, the best instrumentation for determining the required dynamic model statistics is an INS with a data recorder.

10.3.4.6 Reordering Host Vehicle Model States The dynamic coefficient matrices in Table 10.2 are defined for one coordinate axis only, which allows the host vehicle dynamic modeler to use different models for different degrees of freedom. These have been implemented as m-files which return the resulting F (or Φ), Q, and P_0 matrices for a specific axis model.

The resulting state variable order can be transformed to the more conventional order (e.g., as used in inertial navigation error modeling) by an orthogonal matrix. For example, this transformation in the case of the sixth model in Table 10.2 would be

$$\underbrace{\begin{bmatrix} \varepsilon_E \\ \varepsilon_N \\ \varepsilon_U \\ \dot{\varepsilon}_E \\ \dot{\varepsilon}_N \\ \dot{\varepsilon}_U \\ \ddot{\varepsilon}_E \\ \ddot{\varepsilon}_N \\ \ddot{\varepsilon}_U \end{bmatrix}}_{x_d} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}}_{T} \underbrace{\begin{bmatrix} \varepsilon_E \\ \dot{\varepsilon}_E \\ \ddot{\varepsilon}_E \\ \varepsilon_N \\ \dot{\varepsilon}_N \\ \ddot{\varepsilon}_N \\ \varepsilon_U \\ \dot{\varepsilon}_U \\ \ddot{\varepsilon}_U \end{bmatrix}}_{x_a}, \tag{10.29}$$

where the state variable x_a on the right is ordered first by axis, then by derivative order. The state variable x_d on the left is ordered first by derivative order, then by axis.

The transformation matrix T in this case is symmetric and orthogonal, so that

$$T^{-1} = T^{\text{T}} = T.$$

As a consequence, the F, P, and Q matrices of the two types of models are related by the formulas

$$F_d = TF_aT$$
$$P_d = TP_aT$$
$$Q_d = TQ_qT$$
$$F_a = TF_dT$$
$$P_a = TP_dT$$
$$Q_a = TQ_dT$$

because T is a symmetric orthogonal matrix.

This trick is used in the error analysis m-files to convert the per-axis model parameter matrices generated by the m-file `DAMP3ParamsD.m` (used for the figure-8 track dynamic model) into the more conventional order by derivative degree.

10.3.4.7 Clock Error Models Most receiver clocks are relatively inexpensive quartz clocks which are quite stable over periods of time in the order of 0–10 s. This works 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 measurement update period for GNSS receivers is generally in the order of 1 s. This allows the receiver to track its own clock errors relatively accurately, which allows receiver designers to use less-expensive clocks.

Clock Phase and Frequency Tracking The most common receiver clock frequency and phase tracking implementation uses the *type 2 tracker* from Table 10.2 to keep the receiver clock synchronized in phase and syntonized in frequency 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.

Clock Dynamic Model in Continuous Time The clock state vector is then two dimensional, with its dynamic model in continuous time being

$$x = \begin{bmatrix} C_b \\ C_d \end{bmatrix} \tag{10.30}$$

$$\dot{x} = \underbrace{\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}}_{F} x + w(t) \tag{10.31}$$

$$w(t) \stackrel{\text{def}}{=} \begin{bmatrix} w_b(t) \\ w_d(t) \end{bmatrix} \tag{10.32}$$

$$Q_t \stackrel{\text{def}}{=} \mathrm{E}_t\langle w(t) w^{\mathrm{T}}(t) \rangle \tag{10.33}$$

$$= \begin{bmatrix} q_{tbb} & 0 \\ 0 & q_{tdd} \end{bmatrix}. \tag{10.34}$$

That is, the zero-mean white noise processes $w_b(t)$ and $w_d(t)$ are uncorrelated.

This model is a short-term approximation to what is called *flicker noise* in clocks. The power spectral density 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, however.

Clock Dynamic Model in Discrete Time The equivalent model in discrete time will be

$$x_k = \Phi(\Delta t) x_{k-1} + w_{k-1} \tag{10.35}$$

$$\Phi(\Delta t) \stackrel{\text{def}}{=} \exp\ (F\ \Delta t) \tag{10.36}$$

$$= \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \tag{10.37}$$

$$w_{k-1} \stackrel{\text{def}}{=} \begin{bmatrix} w_{b,\ k-1} \\ w_{d,\ k-1} \end{bmatrix} \tag{10.38}$$

$$Q_{k-1} \stackrel{\text{def}}{=} \mathrm{E}_k\langle w_{k-1} w_{k-1}^{\mathrm{T}} \rangle \tag{10.39}$$

$$= \Phi(\Delta t) \left[\int_0^{\Delta t} \Phi^{-1}(s) Q_t(s) \Phi^{-\mathrm{T}}(s)\ ds \right] \Phi^{\mathrm{T}}(\Delta t) \tag{10.40}$$

$$= \begin{bmatrix} q_{tbb}\Delta t + q_{tdd}\Delta t^3/3 & q_{tdd}\Delta t^2/2 \\ q_{tdd}\Delta t^2/2 & q_{tdd}\Delta t \end{bmatrix}, \tag{10.41}$$

where Equation 10.40 is from Equation 4.131, 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

$$P_k = \Phi(\Delta t) P_{k-1} \Phi^{\mathrm{T}}(\Delta t) + Q_{k-1} \tag{10.42}$$

$$Q_{k-1} = \begin{bmatrix} q_{bb} & q_{bd} \\ q_{db} & q_{dd} \end{bmatrix} \tag{10.43}$$

$$q_{bb} = q_{tbb}\Delta t + q_{tdd}\Delta t^3/3 \tag{10.44}$$

$$q_{bd} = q_{tdd}\Delta t^2/2 \tag{10.45}$$

$$q_{db} = q_{bd} \tag{10.46}$$

$$q_{dd} = q_{tdd}\Delta t. \tag{10.47}$$

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 time period equal to the intersample period ($\approx$1 s). The same method can be used for any RMS relative frequency stability and intersample period.

Example 10.1 (Calculating Clock Model Disturbance Noise) Reasonably low cost quartz crystal clocks have frequency stabilities in the order of 10^{-9} to 10^{-6} parts per part over the time between GPS pseudorange measurements (1 s). That is, for clock frequency f, the RMS incremental change in relative frequency $[f(t) - f(t-1)]/f(t-1)$

$$\frac{\sigma_f}{f} \approx 10^{-9}\text{to}10^{-6}. \tag{10.48}$$

We will use the lower value (10^{-9}) to demonstrate a methodology for translating this number to 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

$$Q_{10^{-7}} = \left(\frac{10^{-7}}{10^{-9}}\right)^2 Q_{10^{-9}}, \tag{10.49}$$

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

$$q_{dd} = (c\ \sigma_f/f)^2 \tag{10.50}$$

$$\approx (3 \times 10^8 \times 10^{-9})^2 \tag{10.51}$$

$$\approx 0.1(\text{m}^2/\text{s}^2) \tag{10.52}$$

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

$$q_{tdd} = q_{dd}/\Delta t \tag{10.53}$$

$$= 0.1(\text{m}^2/\text{s}^3). \tag{10.54}$$

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

$$q_{tbb}\Delta t \approx q_{tdd}\Delta t^3/3 \tag{10.55}$$

$$q_{tbb} \approx q_{tdd}\Delta t^2/3 \tag{10.56}$$

$$\approx 0.03(\mathrm{m}^2/\mathrm{s}) \tag{10.57}$$

$$Q_t \approx \begin{bmatrix} 0.03 & 0 \\ 0 & 0.10 \end{bmatrix}, \tag{10.58}$$

and

$$q_{bb} = q_{tbb}\Delta t + q_{tdd}\Delta t^3/3 \tag{10.59}$$

$$\approx 0.06(\mathrm{m}^2) \tag{10.60}$$

$$q_{bd} = q_{tdd}\Delta t^2/2 \tag{10.61}$$

$$\approx 0.05(\mathrm{m}^2/\mathrm{s}) \tag{10.62}$$

$$Q_{k-1} \approx \begin{bmatrix} 0.06 & 0.05 \\ 0.05 & 0.10 \end{bmatrix} \tag{10.63}$$

in distance and velocity units.

A plot of clock estimation uncertainties versus clock stability is shown in Figure 10.7 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.

Clock Measurement Sensitivity Matrix The sensitivity of any pseudorange ρ to the clock state vector has the form

$$H_{\text{clock}} = \frac{\partial \rho}{\partial\ C_b,\ C_d} \tag{10.64}$$

$$= \begin{bmatrix} -1 & 0 \end{bmatrix}, \tag{10.65}$$

when C_b has units of distance. That is, a clock bias error ε_b is equivalent to a uniform increase of ε_b meters in *all* pseudoranges simultaneously.

10.3.4.8 Uncompensated Iono Delay Error Models These are not likely to require further compensation for dual-frequency receivers, but they are a dominant error source for single-frequency receivers.

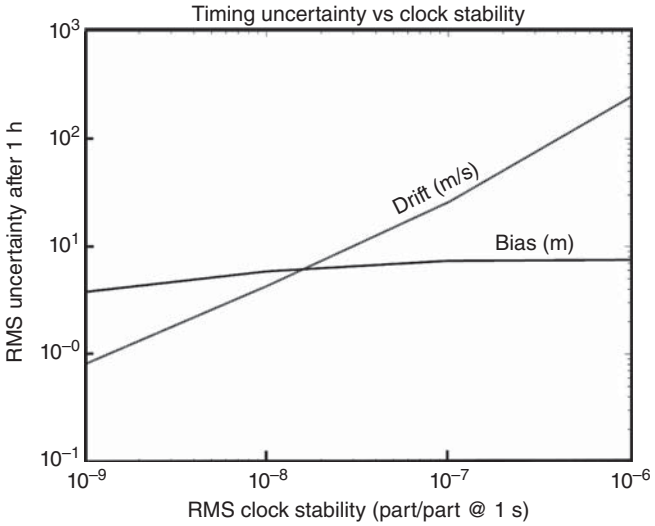


Figure 10.7 Clock bias (synchronization) and drift (syntonization) uncertainties.

Models for Klobuchar Correction Residuals For GPS receivers without two-frequency capability or differential correction, a coarse 3D tomographic model for computing local delay corrections is transmitted as part of the signals from the satellites (see Section 10.3.3.2).

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 psuedorange errors in each satellite signal. The delays are approximated as exponentially correlated processes, the model for which in continuous time is

$$\dot{\delta}_{\text{iono},j} = -\delta_{\text{iono},j}/\tau + w(t) \tag{10.66}$$

$$w(t) \in \mathcal{N}(0, Q_t)$$

$$Q_t = 2\sigma^2/\tau \tag{10.67}$$

$$\sigma \approx 10\text{m},$$

$$\tau \approx 60\text{s}.$$

The equivalent pseudorange error model in discrete time is

$$\delta_{\text{iono},j,k(-)} = \Phi\delta_{\text{iono},j,(k-1)(+)} + w_{k-1} \tag{10.68}$$

$$\Phi = \exp\ (-\Delta t/\tau) \tag{10.69}$$

$$\Delta t = \text{discrete time-step}$$

$$\tau \approx 60\text{s}$$

$$
\begin{aligned}
w_k &\in \mathcal{N}(0, Q) \\
Q &= \sigma^2(1 - \Phi^2) \\
\sigma &\approx 10\text{m}.
\end{aligned}
\tag{10.70}
$$

Measurement Sensitivity Matrix 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, it 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 correlation time constants, this implementation does not compromise performance.

10.3.5 Performance Evaluations

The real power of Kalman filtering is in its implementations, which provide the greatest benefit from all the available data. A secondary benefit is that its expected performance can be assessed before it is built—using simulations.

We use simulators of the host vehicle dynamics and GNSS satellite motions to evaluate relative performance of different Kalman filter implementations for GNSS. These are described in the following subsections.

10.3.5.1 Host Vehicle Dynamic 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 Simulators These are described in Section 10.2.7.2.

Racetrack dynamics The statistical parameters used for tuning host vehicle dynamic models of Table 10.2 are summarized in Table 10.1.

10.3.5.2 Efficacy of Vehicle Dynamic Models Figure 10.8 is a plot of simulated GNSS navigation CEPs as a function of time over an 8-h simulation of dynamics on a 100-km figure-8 test track, with GNSS signals being lost for a minute near the end of each hour. The solid line plots the CEP derived from the covariance matrix of estimation uncertainty, which is the solution of a Riccati equation including a stochastic model for host vehicle dynamics. The host vehicle dynamic model is the sixth one in

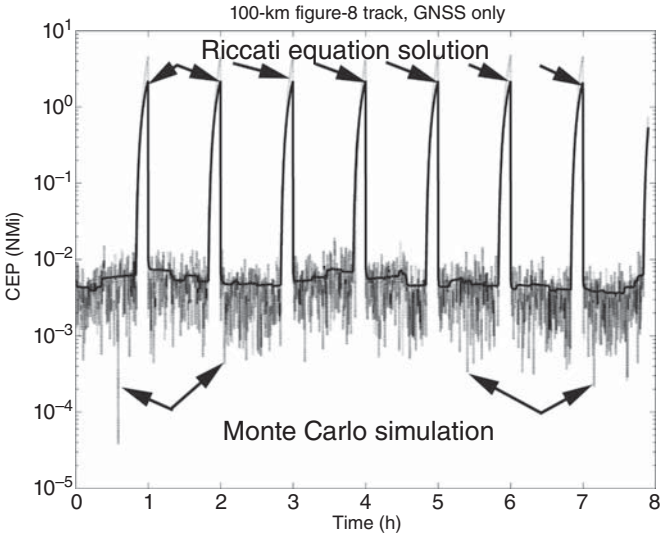


Figure 10.8 Simulated and theoretical performance of GNSS navigation.

Table 10.2, with parameter values taken from Table 10.1. The only model the Kalman filter has for vehicle dynamics is a stochastic one.

The lighter dotted line plots a sample CEP derived from a Monte Carlo simulation with appropriate pseudonoise sources and with the vehicle dynamic conditions dictated by the simulated racetrack geometry.

Agreement between the two solutions is encouraging, given that the Kalman filter model for host vehicle dynamics is based solely on the empirical statistics of the simulator, whereas the Monte Carlo analysis uses the deterministic racetrack dynamic simulator from which the host vehicle stochastic model parameters were derived. In both cases, the CEP values during periods of signal availability are about 10-m RMS.

It is possible to derive a stochastic dynamic model for a vehicle on a test track with only lateral and longitudinal motion, and one would expect better performance with the more accurate dynamic model. However, it is quite remarkable how well a model based on just statistical parameters can do.

10.3.5.3 GNSS Simulation Models

Fixed Satellite Locations These models are used in the MATLAB scripts in `ClockStab.m` (to remove the effects of satellite motion) and `SatelliteGeometry.m` (to demonstrate the influence of satellite geometry on performance) on the companion Wiley web site.

"Real-World" Satellite Motions These GNSS simulations used the GPS satellite configuration from March 2014, as downloaded from the US Coast Guard site

`www.navcen.uscg.gov - /ftp/GPS/almanacs/yuma/.`

There were 30 operational GPS satellites at that time. The MATLAB script `ReadYUMAdata.m` is designed to transform the ASCII text files from that site into MATLAB m-files initializing the satellite simulator. The MATLAB script `YUMAdata.m` was generated by `ReadYUMAdata.m`. It contains the GPS ephemeris information of March 2014, 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.

Circular Orbit Assumption To simplify the calculations required, the GPS satellite simulator assumes circular orbits. This level of model fidelity is generally permissible in performance analysis, even though it would not be permissible for obtaining the navigation solution.

10.3.6 Quality of 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 GNSS receivers 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 adjusted occasionally by adding or deleting "leap seconds" to remain phase-locked to the rotation of the earth within $\approx \pm 7.5$ arcsec. But "GNSS time" cannot be subjected to such discrete changes without inflicting the navigation solutions with pseudorange jumps of 299,792,458 m.
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 error, 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.
7. Time-correlated uncompensated pseudorange errors, due principally to smaller-scale variations in ionospheric delay. The RMS magnitude of these

errors is in the order of 10 m, and the correlation times are in the order of a minute, typically. Estimating these nuisance variables requires one additional Kalman filter state variable for each satellite being used in the navigation solution—which can be in 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 become 50 or more.

10.3.6.1 Effects of Satellite Geometry The covariance matrix of state estimation uncertainty provides a better measure of GNSS navigation performance than the DOP calculation of Section 10.3.3.4.

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 10.2 (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, in terms of estimation uncertainty.

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 fixed directions to these four satellites.

The model uses the two-state clock error model from Example 10.4.5 and three antenna position coordinates in locally level coordinates, for a total of five state variables:

$$x^{\mathrm{T}} = \begin{bmatrix} N & E & D & C_b & C_d \end{bmatrix}, \tag{10.71}$$

where

N, north position in meters;
E, east position in meters;
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,

$$Q = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 6 & 5 \\ 0 & 0 & 0 & 5 & 10 \end{bmatrix} \tag{10.72}$$

$$\Phi = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{10.73}$$

We further assume

- RMS pseudorange measurement error is 15 m,
- Initial RMS antenna position uncertainty is 1 km,
- Initial RMS clock bias is 3 km (10 ms),
- Initial RMS relative frequency uncertainty is 30 m/s (10^{-7} part/part),

so that the covariance matrices

$$R = \begin{bmatrix} 225 & 0 & 0 & 0 \\ 0 & 225 & 0 & 0 \\ 0 & 0 & 225 & 0 \\ 0 & 0 & 0 & 225 \end{bmatrix} \tag{10.74}$$

$$P_0 = \begin{bmatrix} 10^6 & 0 & 0 & 0 & 0 \\ 0 & 10^6 & 0 & 0 & 0 \\ 0 & 0 & 10^6 & 0 & 0 \\ 0 & 0 & 0 & 9 \times 10^6 & 0 \\ 0 & 0 & 0 & 0 & 900 \end{bmatrix}. \tag{10.75}$$

The off-diagonal values of R in Equation 10.74 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

$$H = \begin{bmatrix} -\cos\left(\theta_1\right)\cos\left(\phi_1\right) & -\sin\left(\theta_1\right)\cos\left(\phi_1\right) & \sin\left(\phi_1\right) & 1 & 0 \\ -\cos\left(\theta_2\right)\cos\left(\phi_2\right) & -\sin\left(\theta_2\right)\cos\left(\phi_2\right) & \sin\left(\phi_2\right) & 1 & 0 \\ -\cos\left(\theta_3\right)\cos\left(\phi_3\right) & -\sin\left(\theta_3\right)\cos\left(\phi_3\right) & \sin\left(\phi_3\right) & 1 & 0 \\ -\cos\left(\theta_4\right)\cos\left(\phi_4\right) & -\sin\left(\theta_4\right)\cos\left(\phi_4\right) & \sin\left(\phi_4\right) & 1 & 0 \end{bmatrix}. \tag{10.76}$$

These equations are programmed in the MATLAB program `SatelliteGeometry.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 10.9 and 10.10 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 a good satellite geometry, and the second, a 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.

10.3.6.2 Influence of Host Vehicle Models The MATLAB script `GNSSshootout-NCE.m`[7] on the companion Wiley web site compares side-by-side four different host vehicle dynamic models for the same navigation problem, which is a figure-8 track model described in Section 10.2.7 with time-correlated random vehicle velocity variations. The three host vehicle dynamic models are

(a) [MODL3,] which uses model number 3 (type 2 tracker) from Table 10.2 with parameter values from the track simulator statistics. This vehicle model has six state variables.

(b) [MODL5], which uses model number 5 (bounded RMS velocity) from Table 10.2 with parameter values from the track simulator statistics. This vehicle model has six state variables.

(c) [MODL6], which uses model number 6 (bounded RMS position) from Table 10.2 with parameter values from the track simulator statistics. This vehicle model has nine state variables.

[7]The "NCE" stands for "No Clock Errors."

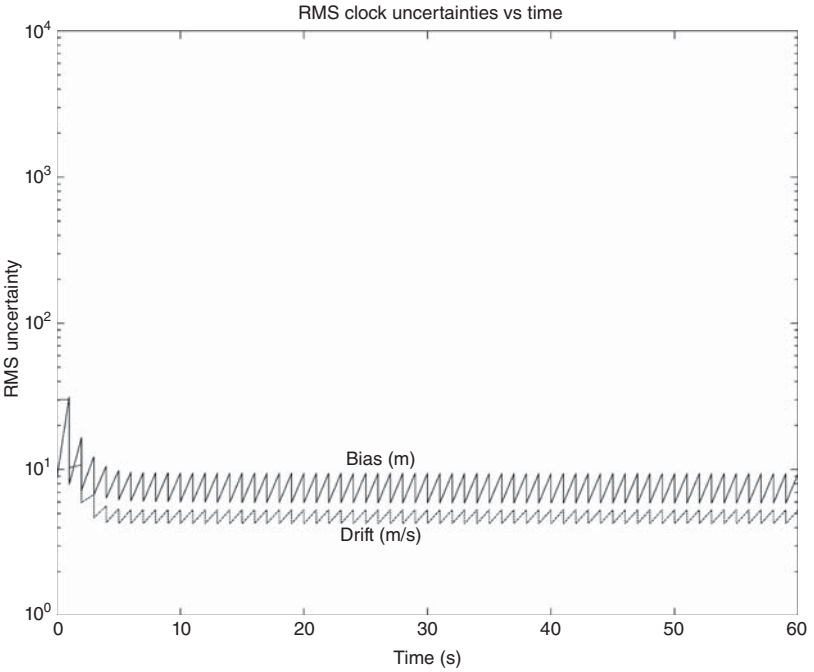


Figure 10.9 Clock parameter uncertainties with good satellite geometry.

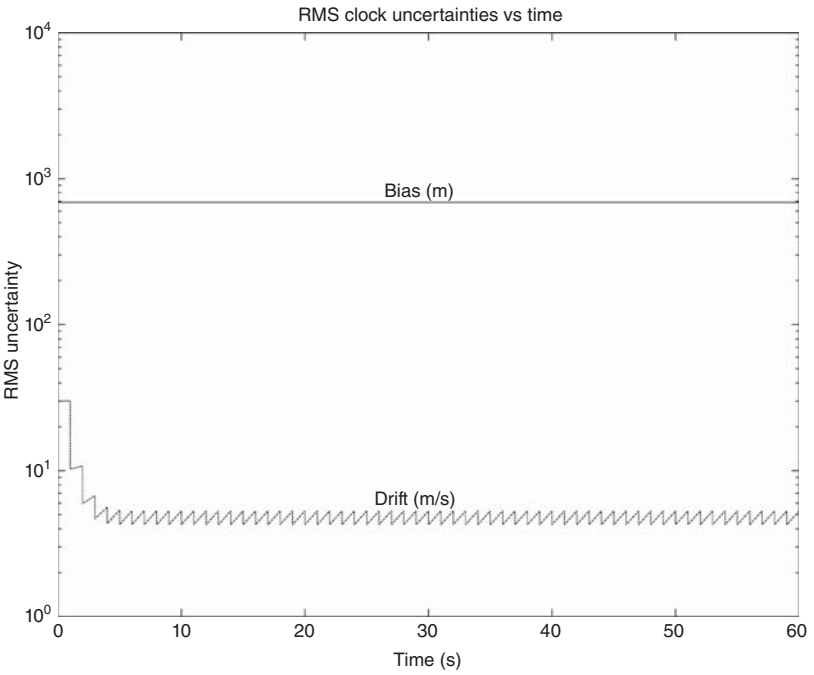


Figure 10.10 Clock parameter uncertainties with bad satellite geometry.

(d) [FIG8], which uses a specialized 1D model for the figure-8 track dynamics, using only along-track distances, velocities, and accelerations. It has three state variables. The model is coded in `GNSSshootoutNCE.m`.

Each model was "tuned" to the simulated host vehicle dynamics by setting its independent model parameters to match the values listed in Table 10.1. 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 10.3.

The values given in Table 10.3 for the number of state variables is for the vehicle dynamic model only. The number of additional state variables required for propagation delay errors was the number of satellites in view ($\geq 15°$ above the horizon). Clock error modeling was not included, but it 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.

10.3.6.3 Single-Frequency Versus Dual-Frequency GNSS Navigation The main difference between single-frequency GNSS navigation and navigation with a dual-frequency receiver (or with differential correction) is that the state variables in the single-frequency case include the uncompensated ionospheric delays, and number of state variables is significantly greater in the single-frequency case. As a consequence, the Kalman filter must expend some of its information in solving for the added state variables, and it has less information left for the navigation variables.

The m-file `GPS2FreqOnly.m` solves the Riccati equation with dynamic coefficient matrix structure like that in the lower left of Figure 10.29, generating plots of the various RMS navigational uncertainties, including the RMS horizontal position uncertainty shown plotted in Figure 10.11.

TABLE 10.3 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	vertical	north	east	vertical
MODL3	6	49.3	34.6	5.7	18.1	16.1	0.4
MODL5	6	27.5	17.2	3.5	20.1	15.0	0.4
MODL6	9	8.4	7.9	3.3	22.2	16.7	0.4
FIG8	3	1.2	0.7	0.02	0.11	0.16	0.002

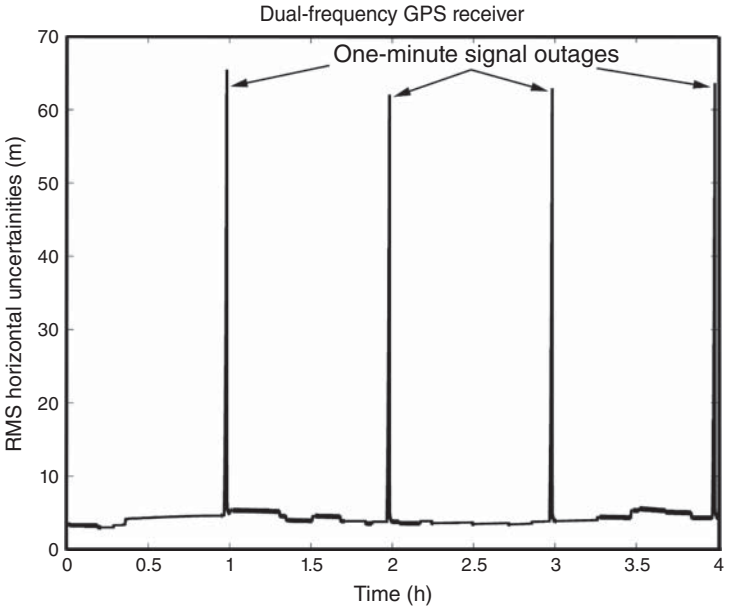


Figure 10.11 RMS horizontal position uncertainty using dual-frequency GPS.

By comparison, the m-file `GPS1FreqOnly.m` solves the Riccati equation with F and Q matrix structures like that in the upper left of Figure 10.29, but with the common parts of the model parameters having the same values. A plot of the resulting RMS horizontal position uncertainty is shown in Figure 10.12.

As expected, the RMS navigation errors are a bit worse for the single-frequency receiver.

10.3.7 Schmidt–Kalman Filtering for Residual Iono Corrections

Single-frequency GPS receivers can use the parametric Klobuchar model built into the GPS satellite message structure for removing much (but not enough) of the ionospheric propagation delays of the received signals. The rest of the correction requires additional Kalman filter state variables for the residual iono errors of each satellite used. This implementation may either assign one state variable for each satellite in the GPS constellation or use a state-swapping strategy to keep just enough state variables for the number of satellites currently being used. Either way, this adds from 4 (minimal solution) to 30+ (maximum solution) nuisance variables to the Kalman filter state vector.

The Schmidt–Kalman filter (described in Chapter 9) is designed to reduce filter computational requirements by ignoring nuisance variables—thereby sacrificing some estimation performance. The resulting filter performance after reducing the number of state variables is taken into account in the associated Riccati equation

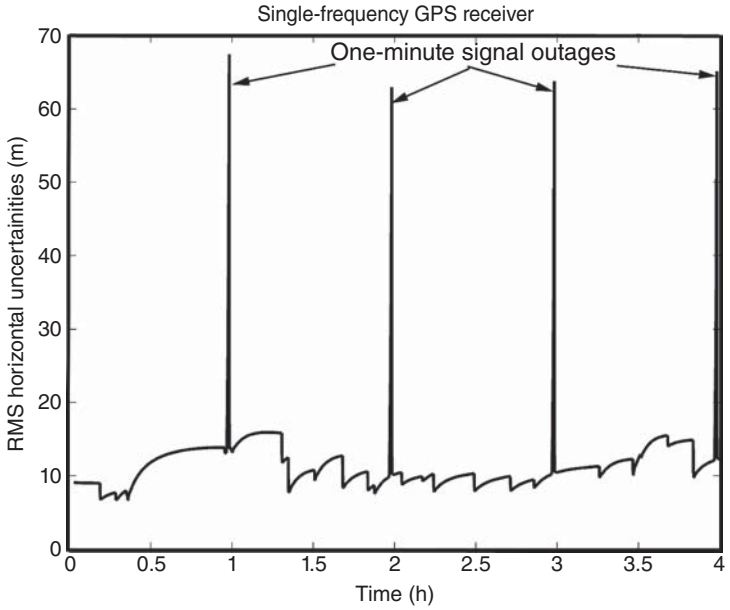


Figure 10.12 RMS horizontal position uncertainty using single-frequency GPS.

implementation, however. This allows the filter designer to assess the performance degradation to be taken into account in the trade-offs between alternative implementations.

These sorts of trade-offs can be important in system processor selection, when the Schmidt–Kalman approach could make the difference between a more expensive processor and a less-expensive but slower alternative.

The MATLAB m-file `SchmidtKalmanTest.m` on the companion Wiley web site evaluates the performance degradation in terms of the penalty in RMS position error added by the Schmidt–Kalman filter during a simulated dynamic test. The alternative Kalman filter implementation in this comparison used one state variable for each satellite in the constellation, although only those more than 15° above the horizon were being tracked. Other details of the simulated conditions are given in the MATLAB script. The result is plotted in Figure 10.13, in terms of the "un-RMSed" difference between the straight Kalman filter implementation and the Schmidt–Kalman implementation,

$$\sigma_{\text{unRMS}} = \sqrt{\sigma_{\text{SKF}}^2 - \sigma_{\text{KF}}^2}.$$

Figure 10.13(a) is the additional position error contribution from the Schmidt–Kalman approximation, and Figure 10.13(b) is the number of satellites being tracked. This shows very well the transient performance degradation each time a satellite is added to or deleted from the list of those being tracked. These transients amount to an additional error contribution in the order of several meters to more than 10 m, with

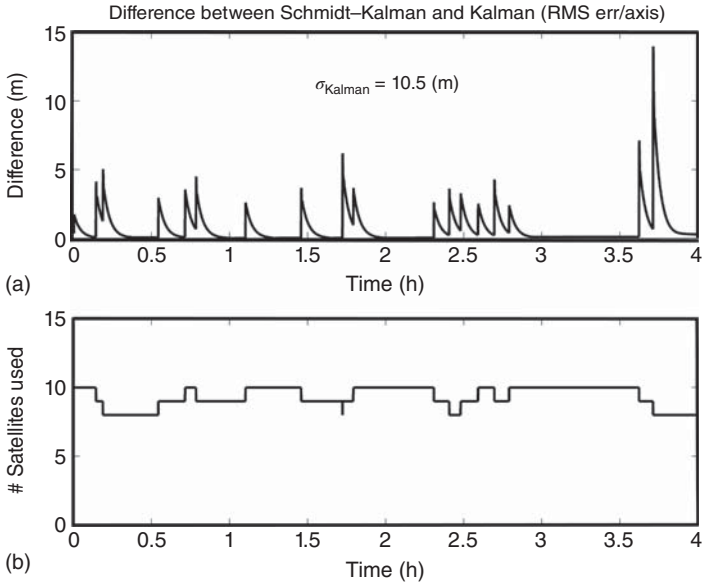


Figure 10.13 Performance Degradation with Schmidt–Kalman filtering.

the transient effect vanishing after several minutes. In this particular example, the degradation is rather small, considering that the baseline Kalman filter navigation performance is in the order of 10-m RMS.

Any decision about whether to use the Schmidt–Kalman filter should be based on the intended implementation conditions. It is also important in making this sort of trade-off decision to quantify the computational savings associated with the performance loss.

10.3.8 Using Pseudorange Differences

By exploiting known error characteristics in the signals, GNSS users have devised the following methods (and many more) to improve GNSS navigation performance. We mention them here to identify alternative methods available to system implementers.

10.3.8.1 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 the differences are relatively insensitive to signal propagation delays, these measurements are useful for estimating host vehicle velocities.

10.3.8.2 Spatial Differences Pseudoranges from separate antennas are also used in GNSS navigation. Receiver antennas at known fixed locations are used to estimate

pseudorange errors for 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 psuedorange measurement accuracies for each satellite. This approach can provide position accuracies in the order of centimeters if carrier phase tracking is used.

10.3.8.3 Derived Attitude Estimates 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 in the order of a meter can provide RMS attitude accuracies of a milliradian or less.

Although GNSS pseudorange measurements at a single antenna are insensitive to antenna attitude, the estimated antenna velocity and acceleration are commonly used to generate secondary estimates of heading, pitch angle and roll angle. Here, these secondary estimates will assume the vehicle roll axis is aligned with the direction of velocity, and that turns are "coordinated." That is, the sensed accelerations are kept quasi-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

$$\tan\left(\widehat{\theta}_Y\right) = \frac{\widehat{V}_E}{\widehat{V}_N}(\text{heading angle}) \tag{10.77}$$

$$\sin\left(\widehat{\theta}_P\right) = \frac{-\widehat{V}_D}{\sqrt{\widehat{V}_N^2 + \widehat{V}_E^2 + \widehat{V}_D^2}}(\text{pitch angle}) \tag{10.78}$$

$$\sin\left(\widehat{\theta}_R\right) = \frac{U_D}{\cos\left(\widehat{\theta}_P\right)}(\text{roll angle}) \tag{10.79}$$

$$U = \frac{\widehat{V} \otimes \left[I - \frac{\widehat{V}\widehat{V}^{\mathrm{T}}}{\widehat{V}^{\mathrm{T}}\widehat{V}}\right]\widehat{A}}{\left|\widehat{V} \otimes \left[I - \frac{\widehat{V}\widehat{V}^{\mathrm{T}}}{\widehat{V}^{\mathrm{T}}\widehat{V}}\right]\widehat{A}\right|}, \tag{10.80}$$

where $\widehat{V}$ is the estimated velocity vector and $\widehat{A}$ is the estimated sensed acceleration vector (i.e., not including gravitational acceleration) in north-east-down (NED) coordinates, and $\widehat{\theta}_Y$, $\widehat{\theta}_P$, and $\widehat{\theta}_R$ are the estimated vehicle heading, pitch, and roll angles.

This model is only good so long as the magnitude of velocity is much greater than the RMS velocity uncertainty.

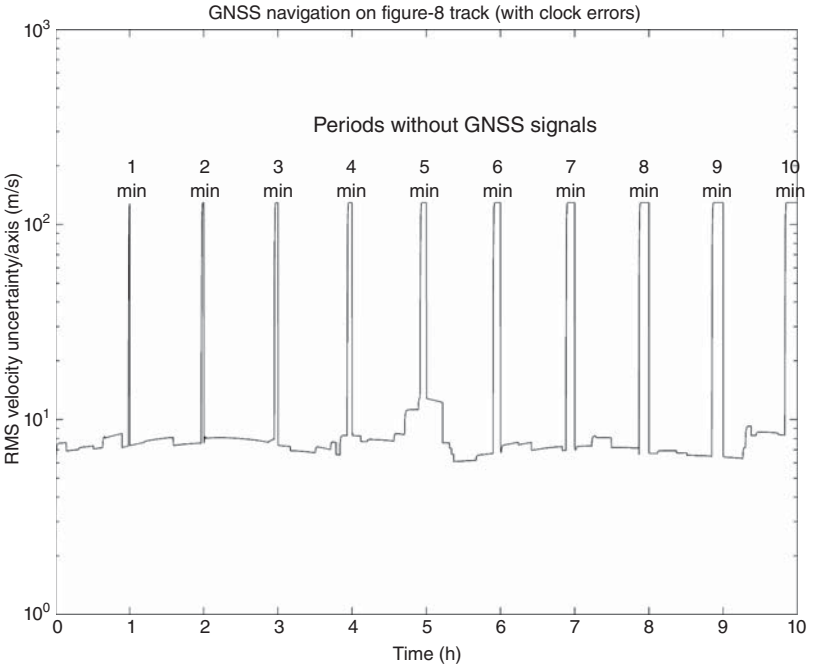


Figure 10.14 RMS velocity uncertainty for GNSS navigation on figure-8 track.

Results from a GNSS-only simulation on a figure-8 track are plotted in Figure 10.14. The simulated velocity magnitude $|V| \approx 25$ m/s and its standard deviation $\sigma_V \approx 7.7$ 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 in the order of $\sigma_\theta \approx 0.3$ rad $\approx 17°$. This is much worse than the short-term RMS tilt uncertainties in INS navigation, and 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 $|V| \approx 100$ m/s would have attitude uncertainties in the order of 4–5. For high performance aircraft with $|V| \approx 300$ m/s, the RMS attitude uncertainties in this approximation would be in the order of 1–2. 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.

10.4 INERTIAL NAVIGATION SYSTEMS (INS)

The purpose of this section is to derive, implement, and demonstrate INS navigation error models for Kalman filters used in GNSS/INS integration. Early military

"stand-alone" (i.e., unaided) inertial navigators did not use Kalman filters for navigation because INS development began before the Kalman filter was available, but inertial navigation and airborne radar would be instrumental in the development of flightworthy computers that would later be useful for Kalman filter implementations.

There are only a couple of operational GNSS systems, with perhaps one or two more on the way. Their error models at the receiver level have a lot in common.

Inertial navigation is far more diverse. There are perhaps hundreds or thousands of different INS designs. Their error models are much more diverse at the GNSS/INS integration level. We will select a few example error models to demonstrate the general principles exploited by Kalman filtering, but keep in mind that these are but a few of the many.

The main problem with inertial navigation is that its errors increase with time unless other navigation sources are used to keep them in check. This is not necessarily a serious problem for ballistic missiles, because their time of active inertial navigation is only for the few minutes of launch. But it is a problem for inertial navigation of land vehicles, aircraft, and ships.

Soon after the introduction of the Kalman filter, it was used to assess the relative advantages of using various forms of "external" navigation solutions (including GNSS) for keeping INS errors in check. Models developed for these studies would play a major role in the initial design, development, and implementation of satellite-based navigation systems.

10.4.1 A Little Background

Inertial navigation has also been called *Newtonian navigation*. The basic idea comes from Newton's calculus: *The second integral of acceleration is position.* Given that and sensors capable of measuring the three components of acceleration over time, plus initial values for position and velocity, turning this into a navigation system[8] would appear to be a relatively straightforward integration problem. Reducing it to practice would prove to be a bit more complicated, however.

The oldest known INSs are of the type you carry around in your head. There are two of them, and they are part of the *vestibular systems* located in the bony mass behind each ear. They have been evolving since the time your ancestors were fish. Each includes three rotational acceleration sensors (*semicircular canals*) and two 3-axis accelerometers (*otolith organs*). Their primary function is to compensate your vision system when you move your head. They are not accurate enough for long-distance navigation, but they do enable you to balance and navigate in total darkness for short distances.

10.4.1.1 History Enabling technologies for practical long-distance inertial navigation would not emerge until the twentieth century, at the beginning of which precision

[8] German entrepreneur Johann M. Boykow (1878–1935) was one who foresaw the possibility and led much of the pioneering German pre-WWII inertial sensor development at the gyroscope company Kreiselgeräte GMBH in Berlin.

gyroscopes were being developed to replace the magnetic compass—which was not very useful aboard iron ships.

After the 1919 Treaty of Versailles had forbidden long-range artillery development in Germany, efforts there shifted to longer-range cruise missiles (V-1) and ballistic missiles (V-2). These require guidance and control systems to direct them to their targets, which would be done using rotation sensors (gyroscopes) and acceleration sensors (accelerometers). Rotation sensors were used to keep track of where the acceleration sensors were pointing relative to their intended targets.

The end of World War II (WWII) saw the beginning of the "atomic age," the Cold War between the Soviet Union and NATO allies, and the race to develop long-range delivery systems for nuclear weapons. Building on German WWII experience (and with the aid of some of its personnel), inertial navigation development in the United States and the Union of Soviet Socialist Republics was rather rapid. Much of this development was (and still is) highly classified, so technical details are not always easily come by.

By 1953, an MIT-designed INS[9] could successfully navigate a WWII-era B-29 bomber from Bedford, Massachusetts to Los Angeles, California. The INS alone weighed more than a tonne and was about the size of a Smart car. By the time of the dissolution of the Soviet Union in 1991, the technology for inertial navigation had become quite mature. Inertial systems were then very much more accurate and much smaller but still quite expensive.

At about that time (the early 1990s), the United States was transitioning from its first-generation satellite navigation system (Transit) to its second-generation system (GPS) Both had been developed for integration with inertial navigation systems, which greatly increases INS operational time and range, and reduces inertial sensor performance requirements for achieving a given level of navigation accuracy.

Inertial navigation systems were initially designed as stand-alone, self-contained, secure navigation systems that need not rely on any external aids except for initialization—which was ideal for its initial application to guidance and control of long-range ballistic missiles. While stationary, it is capable of self-alignment with respect to Earth-fixed coordinates and self-determination of its latitude but requires input of its initial longitude and altitude from other sources.

When the Kalman filter first appeared, it first replaced most of the methods used for calibrating inertial sensors and for initial self-alignment. However, it was soon applied to using any available auxilliary information to improve INS functionality and performance. This came to include airborne radar (for aircraft) and (for high altitude flight) star trackers. It also came to include using Transit, LORAN, and GPS for navigation aiding and using sensed acceleration and rotation matching for initial alignment with respect to a carrier-borne INS.

Development of MEMS technologies in the 1970s and 1980s had significantly reduced the cost of inertial sensors at the lowered performance levels required for integrated GNSS/INS navigation. MEMS sensors are not only inexpensive but also

[9]Developed under technical leadership by INS pioneer Charles Stark Draper (1901–1987) and named "Space Inertial Reference Equipment," (SPIRE).

extremely small and light and require very little power. These attributes have combined to enable a significant expansion of the applications markets for integrated GNSS/INS systems.

10.4.1.2 Basic Concepts

Inertia is the propensity of rigid bodies to maintain constant translational and rotational velocity, unless disturbed by forces or torques, respectively (Newton's first law or motion).

An *inertial reference frame* is a coordinate frame in which Newton's laws of motion are valid. Inertial reference frames are neither rotating nor accelerating. They are not necessarily the same as the *navigation coordinates*, which are typically dictated by the navigation problem at hand. For example, "locally level" coordinates used for navigation on or near the surface of the earth are rotating (with the earth) and accelerating (to counter gravity). Such rotations and accelerations must be taken into account in the practical implementation of inertial navigation.

Inertial sensors measure inertial accelerations and rotations, both of which are vector-valued variables.

- *Gyroscopes* (commonly shortened to "gyros") are sensors for measuring rotation. *Rate gyros* measure rotation rate, and *displacement gyros* (also called *whole-angle gyros*) measure accumulated rotation angle. Inertial navigation depends on gyros for maintaining knowledge of how the accelerometers are oriented in inertial and navigational coordinates. There are many basic physical mechanisms for sensing rotation, including
  - Momentum-wheel gyroscopes use the law of conservation of angular momentum (in direction and magnitude) of a spinning mass to establish an inertial reference direction—the rotation axis of the spinning mass. The Foucault gyroscope is a momentum-wheel gyroscope.
  - Lightwave gyroscopes, which use the constant speed of light in opposite directions around a closed path to detect rotations in the plane of that path. Two basic designs are
    - Laser gyroscopes, in which the light is propagated through lasing cavities and reflected off mirrors to complete the loop. Rotations are detected by interferometry of the relative phases in the counter-rotating light paths. Laser gyroscopes are rate integrating.
    - Fiber-optic gyroscopes (FOG), in which light from a laser source is coupled into an optical fiber with many windings around a closed path. Rotation is detected using interferometry between the laser source and the light exiting the loop, based on a phenomenon called the *Sagnac effect.*
  - Vibrating Coriolis gyroscopes , which detect the out-of-plane forces on a body vibrating in a plane due to rotation of the trajectory (Coriolis effect). There are many different designs. "Tuning fork" gyroscopes, for example,

detect rotational vibrations of a tuning fork due to the Corilis effect on the vibrating tines. Hemispherical resonator gyroscopes (also called *wine glass* gyros) detect shifts in the phasing of mechanical vibrational modes caused by the Coriolis effect during rotation. A microscale MEMS design by the Charles Stark Draper Laboratory (now Draper Laboratory) detects out-of-plane motions of a vibrating free layer of silicon supported by silicon "wires" and powered by an electrostatic "comb drive" developed at the Berkeley Sensor and Actuator Center (BSAC) of the University of California.

- *Accelerometers* are sensors for measuring inertial acceleration, also called *specific force* to distinguish it from what we call "gravitational acceleration." *Accelerometers do not measure gravitational acceleration*, which is perhaps more accurately modeled as a warping of the space-time continuum in a gravitational field. An accelerometer in free fall (e.g., in orbit) in a gravitational field has no detectable input. What accelerometers measure is modeled by Newton's second law as $a = F/m$, where F is the physically applied force (not including gravity), m is the mass it is applied to, and *specific force* is the ratio F/m. This all has a profound effect on inertial navigation error propagation. Basic types of acceleration sensors include
  - Pendulous integrating gyroscopic accelerometers (PIGA[10]) in which the center of support of a momentum-wheel gyroscope is offset axially from its center of mass, which creates a precession torque proportional to the component of acceleration orthogonal to the mass offset. These are *integrating accelerometers*, because the output accumulated precession angle is proportional to the integral of input acceleration. They are also the most accurate (and expensive) accelerometers.
  - Proof-mass accelerometers, which measure the force necessary to keep a mass centered in its instrument housing. There are many methods for measuring this force, leading to many different accelerometer designs.
- The *input axis* of an inertial sensor defines which vector component of acceleration or rotation rate it measures. Multiaxis sensors measure more than one component.

An *inertial sensor assembly* (ISA) is an ensemble of inertial sensors rigidly mounted to a common base to maintain the same relative orientations, as illustrated in Figure 10.15. Inertial sensor assemblies used in inertial navigation contain at least three accelerometers (represented by blocks with "A" on each face) and three gyroscopes (represented by blocks with "G" on each face), as shown in the figure, but they may contain more and they may use one multiaxis sensor in place of an equivalent number of single-axis sensors. Reasons for using more than three sensor input axes for an ISA include

[10]Invented by Fritz K. Mueller (1907–2001) in Germany in the 1930s and brought (with Mueller) to the United States in 1945. It was originally used in Germany for controlling ballistic missile range by integrating the total sensed thrust acceleration and signaling for engine cut-off when a predetermined range was attained. It soon became the dominant high accuracy accelerometer design.

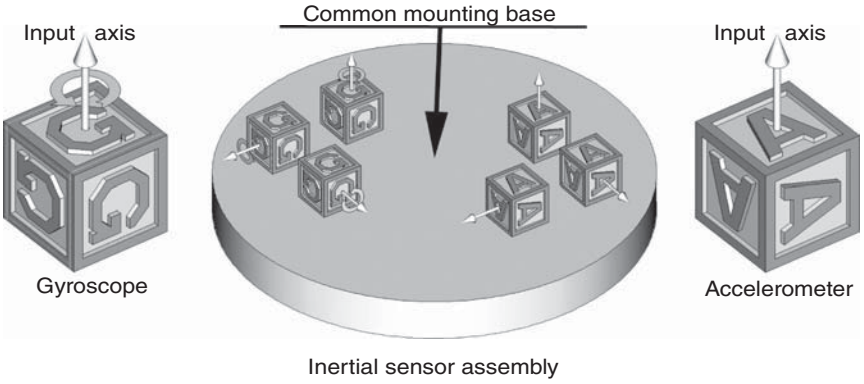


Figure 10.15 Inertial sensor assembly (ISA) components.

- Added sensor redundancy for coping with single-senor failures.
- Reducing effective sensor errors in certain input directions.

The term *inertial reference unit* (IRU) sometimes refers to an inertial sensor system for attitude information only (i.e., using only gyroscopes). Other terms used for the ISA are *instrument cluster* and (for gimbaled systems) *stable element, inertial platform,* or *stable platform.*

An *inertial measurement unit* (IMU) includes an ISA and its associated support electronics for calibration and control of the ISA (possibly including temperature control). These can be small enough to be packaged in a wristwatch.

An *inertial navigation system* (INS) consists of an IMU plus the following:

- *Navigation computers* (one or more) to calculate the gravitational acceleration (not measured by accelerometers) and process the outputs of the accelerometers and gyroscopes from the IMU to maintain an estimate of the position of the IMU. Intermediate results of the implementation method usually include estimates of velocity, attitude, and attitude rates of the IMU.
- *User interfaces*, such as display consoles for human operators and analog and/or digital data interfaces for vehicle guidance[11] and control functions.
- *Power supplies* and/or raw power conditioning for the complete INS.

Some INS designs include temperature control.

Navigation Reference An INS estimates the position of its ISA, just as a GNSS receiver estimates the position of its antenna. When these two systems are integrated together, the offset between these two reference points (ISA and antenna) must be taken into account.

[11] "Guidance" generally includes the generation of command signals for controlling the motion and attitude of a vehicle to follow a specified trajectory or to arrive at a specified destination.

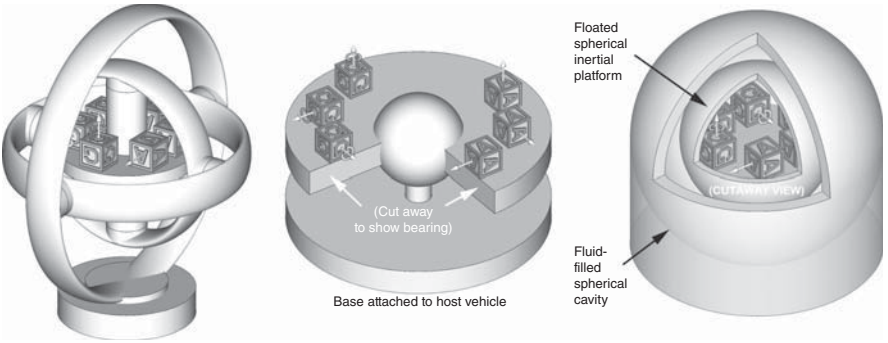


Figure 10.16 ISA attitude isolation methods. (a) Gimbal (cardian suspension), (b) ball joint ("inverted gimbals"), and (c) floated sphere ("flimbals").

10.4.1.3 Sensor Configurations There are two basic configurations for implementing inertial navigation, and some variations within each of the following:

- *Gimbaled systems* use gimbals (also called a *Cardan*[12] *suspension*) to isolate the ISA from inertial rotations or equivalent configurations as shown in Figure 10.16. The orientation of the ISA may also be controlled to align it with inertial coordinates, navigation coordinates, or rotating navigation coordinates. For navigation in the near-Earth environment, these may include locally level orientations in which one ISA axis is slaved to point in the direction of the local vertical (gravity), leaving the other two axes free to assume directions such as
  - One axis east and one axis north (impossible at the poles).
  - An "alpha-wander" system with a known angle α between a locally level axis and an Earth-fixed direction, which solves the problem at the poles.
  - Orientations rotating about the vertical, either continuously ("carouseling") or discretely ("indexing"), either of which cancels out certain types of navigation errors.
- *Strapdown systems* , in which the ISA is "quasi-hard" mounted (i.e., with possibly some shock and vibration isolation) to the carrier vehicle. These systems essentially have "software gimbals" using the gyro outputs to keep track of the orientation of the ISA at all times. An incentive for strapdown configurations is that software is generally much less expensive than hardware. MEMS inertial navigators are generally strapdown, and some designs allow more than one independent and orthogonal input axes in the plane of the MEMS substrate.

[12]Named for Gerolamo Cardano (1501–1576), a Renaissance polymath who did not claim he invented gimbals. The earliest known descriptions of gimbals date from the third century BCE, in Greece and in China.

- Hybrid strapdown systems with a single gimbal axis for rotating the ISA (either continuously or discretely) about a "near-vertical" axis. (Most manned vehicles, e.g., try to keep their passengers oriented with their tall axes near vertical.) This has nearly the same benefits as carouseling or indexing, without the additional cost of other gimbals.

10.4.2 Navigation Solution

As a minimum, any navigation solution must include the current position of the host vehicle in navigation coordinates. For inertial navigation, the intermediate results must also include velocity and attitude. These ancillary variables may also be useful for other aspects of the mission, including

- Attitude, which is useful for driving aircraft cockpit displays for heading (compass card), and roll and pitch angles (artificial horizon). It is also useful for determining lateral wind speed (e.g., for decrabbing in cross-wind landings).
- Attitude rate, which can be used in the control loops for autopilots or missile steering, and for aiding antenna switching and phase locking for navigation satellite signals in GNSS/INS integration.
- Velocity, which is useful for determining time of arrival, and in control loops for autopilots and targeting. It can also be used to compensate for Doppler shift during GNSS signal acquisition and tracking.
- Acceleration, which is useful as an input to control loops for vehicle guidance and control, and as an aid in phase tracking of satellite signals in GNSS/INS integration.

10.4.2.1 Gimbaled Implementation For gimbaled systems, the attitude solution is maintained by the gimbals, which must include gimbal angle encoders for determining the relative attitude of the host vehicle.

The fundamental processing functions for a gimbaled INS are shown as a flowchart in Figure 10.17(a). The boxes in this flowchart represent the major processing functions, including

- The function of the gimbals to keep the input axes of the inertial sensors at known directions in navigation coordinates. In this particular configuration, the accelerometers and gyroscopes have their input axes pointed east, north, and up. The gimbal angles (θ_j) are used for driving attitude displays.
- Integration of acceleration to get velocity.
- Integration of velocity to get position.
- Compensation of the sensor outputs for errors determined during calibration.
- Generation of predicted gravitational acceleration of navigation coordinates (locally level).
- Generation of predicted locally level coordinate rotation rates and Coriolis accelerations for velocities over a curved Earth surface.

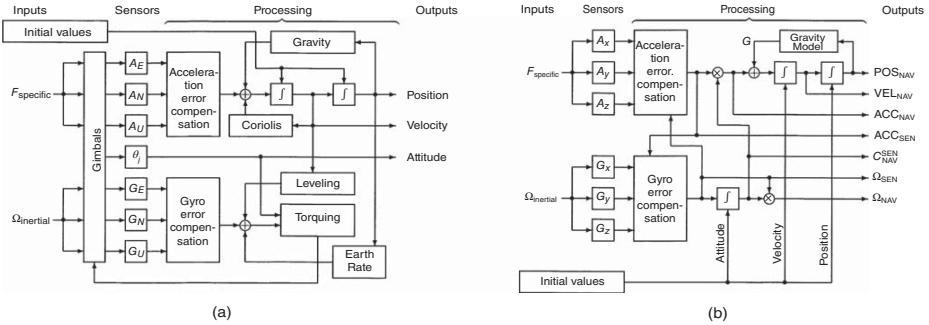


Figure 10.17 INS data flow. (a) Gimbaled and (b) strapdown.

- Compensation for Earth rotation rate.
- Compensating for the Coriolis effect in rotating coordinates.
- Generation of gimbal torquing commands for maintaining the ISA aligned with navigation coordinates.

Acceleration and attitude sensors in gimbaled systems are isolated from large rotation rates, which has some advantages in sensor design.

10.4.2.2 Strapdown Implementation For strapdown systems, attitude information is maintained in memory, and there are no gimbals to isolate the inertial sensors from rotations of the host vehicle. This eliminates the use of gyroscopic (e.g., PIGA) accelerometers, because they are also rotation-sensitive. It also places greater demands on rotation sensors to operate over wider dynamic ranges. Otherwise the fundamental implementation functions are much the same as in gimbaled systems.

The fundamental processing functions are shown in Figure 10.17(b).

The fundamental differences between gimbaled and strapdown implementations are

1. Gimbals are a hardware solution to the attitude problem. For strapdown systems, the solution must be implemented in the software.
2. Strapdown systems expose their sensors to much greater dynamic ranges of rotation rates, which places greater demands on their design—and on the algorithms used for maintaining the "attitude solution."
3. The "attitude solution" is characterized by $C_{\text{SEN}}^{\text{NAV}}$, the coordinate transformation matrix from sensor-fixed coordinates to navigation coordinates. A factor of $C_{\text{SEN}}^{\text{NAV}}$ is the only difference between the error models for strapdown and gimbaled systems.

10.4.3 Initializing the Navigation Solution

As mentioned at the beginning of Section 10.4.1, inertial navigation requires initial values for position, velocity, and attitude of its ISA. Historically, some of this was done using only internal resources—if the ISA could be kept motionless with respect to the earth's surface for several minutes to a few hours (depending on the accuracy required). When the Kalman filter came along in the early 1960s, methods were developed to allow initialization "on the fly" in much less time.

10.4.3.1 Gyrocompass Alignment This is an "almost complete" initialization method using only the inertial sensors.

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 could be done with the INS held motionless with respect to the surface of Earth, in which case its initial velocity in

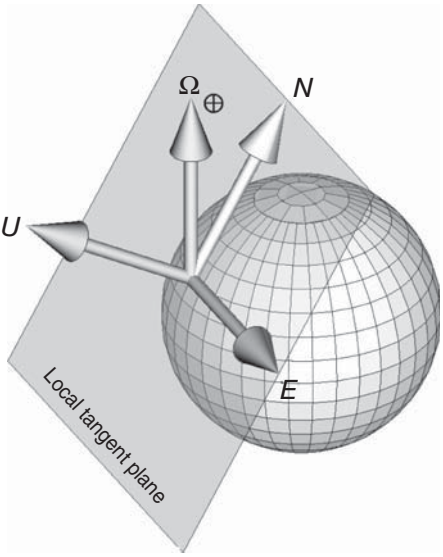


Figure 10.18 Gyrocompass alignment geometry.

Earth-fixed coordinates is zero and its position is known. That leaves the problem of determining its orientation. Figure 10.18 illustrates how it is done:

1. The direction of sensed acceleration is upward, which establishes that direction in ISA coordinates.
2. Except at the poles, the sensed Earth rotation vector has a north component. That is sufficient to establish the orientation of the ISA with respect to east-north-up (ENU) coordinates. Gyrocompass alignment does not work at the poles.
3. The angle between the vertical and the Earth rotation axis equals 90° minus latitude, from which latitude can be determined.

The only missing initial conditions are longitude and altitude. Given latitude and the magnitude of sensed acceleration, one might get a very crude estimated of altitude—but generally not good enough for navigation. Before GNSS or its equivalent, these data had to be entered manually or from local wireless sources.

Gimbaled Implementation Alignment of a gimbaled INS while the ISA remains in a stationary position with respect to Earth can be implemented by rotating the gimbals to force the outputs of the east accelerometer, north accelerometer, and east gyro to be zero. These three constraints are sufficient to solve the alignment problem—except at the poles.

Milli Earth Rate Units (MERUs) The gyroscopes in gimbaled systems are primarily used for maintaining the ISA aligned with navigation coordinates. As such, the dominating input rate will be the Earth rotation rate of about 15° per hour. At that rate, a relative scale factor error of 10^{-3} would be equivalent to 0.015/h, or 1 " MERU" (an acronym for "milli Earth rate unit"), defined at the MIT Instrumentation Laboratory (now the Draper[13] Laboratory) as being about the gyro bias accuracy required for gyrocompass alignment to 1 mrad of accuracy.

Gyro Bias and Scale Factor Requirements Gyro bias is the gyro output offset error. Gyro bias requirements for medium accuracy navigation (1 NMi/h CEP rate) are usually specified to be significantly less than 1 MERU, in the order of 10^{-3} to 10^{-2} degrees/h, and scale factor requirements for gimbaled systems are often specified to be in the order of 10^{-4} parts per part (just to add a little cushion). This has given some much-needed relief in scale factor requirements for gyros in gimbaled systems. Scale factor requirements for that level of performance in strapdown systems, on the other hand, will depend on the expected body rotation rates of the host vehicle, which can be hundreds of degrees per second—or millions of MERUs. In that case, scale factor stability requirements for strapdown inertial navigation could be a millionth of that required for gimbaled systems. Because of this, the savings from eliminating gimbals in strapdown systems can be partially eaten up by the added cost of better gyroscopes.

Kalman Filter Implementations Gyrocompass alignment is a bit more difficult aboard parked aircraft and docked ships, which can be subjected to external disturbances due to winds, tides, and waves, and to internal disturbances during fueling, cargo loading, or passenger loading. In these cases, gyrocompass alignment can still be done using a Kalman filter with a stochastic dynamic model for these disturbances. These generally include damped harmonic oscillator models tuned to match the support or suspension systems (including rotational dynamics) of the host vehicle during such operations.

Initializing Position Although gyrocompass alignment can estimate INS latitude, it has become 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 under way. The GNSS solution also estimates the direction of travel, which can be used as an initial estimate of heading.

10.4.3.2 Alignment Using Auxilliary Information There are a number of methods for speeding up alignment by using other sources of information, such as

[13]When asked to give some general notation of the magnitude of 1 MERU, Charles Stark Draper is alleged to have replied "It's about how fast you have to twist my arm to get me to take a drink."

1. An INS-equipped vehicle carried by another host vehicle with its own INS can use information from the host INS to initialize its own INS in a Kalman filtering procedure called *velocity matching*. Carrier-based aircraft, for example, can use the sensed roll and pitch of the ship and the known direction of the launch catapult action relative to the ship. This requires a Kalman filter similar in some ways to that used in gyrocompassing, but generally initializes much faster due to the additional information.
2. *Transfer alignment* uses a Kalman filter to initializing the navigation solutions of tactical missiles carried by an aircraft with its own INS, by matching their velocities during maneuvering. The same Kalman filter model can be used offline to select those types of maneuvers most useful for speeding up alignment and for determining when such initialization is adequate for the intended mission.
3. Integrated GNSS/INS navigation is capable of initializing the INS "on the fly," using the GNSS solutions.

10.4.4 INS Navigation Error Sources

Inputs to inertial navigation systems include

- Initial conditions for the navigation solution, including position, velocity, and orientation of its ISA.
- Outputs of acceleration and rotation sensors on the ISA, used for carrying the initial navigation solution forward during dynamic disturbances of the ISA.
- Outputs from internal models for unsensed rotations and accelerations of its navigation coordinates. For navigation with respect to the surface of the earth, for example, these will include Earth rotation rate, gravitational acceleration, and centrifugal forces and Coriolis effects from velocities with respect to a rotating earth.
- Any navigation aids used for keeping inertial navigation errors in check. These may include radio navigation fixes, such as those from navigation satellites.

Each of these is a potential error source, as are any modeling errors and roundoff errors in the computer software implementation.

10.4.4.1 Core Error Variables The minimal number of navigation error variables for keeping track of inertial navigation errors during operation is nine. The "core nine" include

1–3: Three components of position error.
4–6: Three components of velocity error.
7–9: Three components of attitude error.

10.4.4.2 Dynamic Noise Sources One source of effective dynamic disturbance noise in inertial navigation is noise on the outputs of the inertial sensors (gyroscopes and accelerometers), including roundoff errors for digital sensors. These are not treated like sensor noise in a Kalman filter implementation but like dynamic disturbance noise on the core variables of the navigation solution. There are at least six such sources, one for each input axis of each sensor:

- Three or more "angular random walk" noise sources from three or more rate gyroscopes. Even "whole-angle" gyroscopes can exhibit noise, but it is more likely to be modeled as angular white noise.
- Three or more acceleration noise sources from three or more accelerometers.

The respective output noise characteristics can be determined empirically by observing sensor outputs with constant sensor input. If the noise is not white (uncorrelated), then additional state variables may be needed for sensor noise modeling.

10.4.4.3 Sensor Compensation Errors Each of the six (or more) inertial sensors has input/output errors due to the limits of manufacturing precision and calibration uncertainty, and these can be nonconstant errors that drift slowly over time due to intrinsic and environmental factors. They can usually be modeled as "slow variables" such as random walk processes or exponentially correlated random processes, each of which requires adding one state variable to the Kalman filter state vector. The simplest and most common sensor compensation errors are (i) output bias and (ii) input/output scale factor, but there may be other nonlinear sources as well. Just the biases and scale factors alone could total 12 or more in number (2 variables for each of 6 or more sensors), but high precision sensors can have even more error characteristics requiring continual updating from the sensor-integrating Kalman filter.

Gimbaled versus Strapdown Models Although the dynamics of INS navigation errors are modeled here in locally level coordinates, the same dynamic model applies to any INS configuration—gimbaled or strapdown. However, the models for how sensor noise and sensor compensation errors couple into navigation errors will be different for different gimbaled implementations (e.g., ENU versus carouseled) and for strapdown implementations. The models used here and in the next section are for a gimbaled ENU, but their transformations for other implementations is straightforward, as described in Sections 10.4.5.1, 10.4.5.4, and 10.4.6.7.

10.4.4.4 INS Error Budgeting *Error budgeting* is the process of allocating subsystem performances in order to achieve some specified system-level performance. For INS, the dominant subsystems in this breakout are the sensors. It is an important part of system design because it allows reallocation of subsystem performance levels to minimize system cost, for example, or to accommodate derating of one component by tightening the performance requirements of others. It has been extremely important in the development of GNSS.

In the case of inertial navigation, system-level performance may be specified in different ways, depending on the application. Performance for intercontinental ballistic missiles, for example, is commonly specified in terms of RMS miss distance or CEP at the target. Performance for applications in cruise navigation, where navigational accuracy tends to deteriorate over time, is commonly specified in terms of how fast RMS navigational uncertainties grow with time. Performance in the latter case is often specified in units of nautical miles per hour CEP rate.

In some cases, the INS error budgeting process may only require evaluating expected performance with different choices of available accelerometers and gyros. In other cases, it may involve determining the required performance specifications for a sensor being developed for a specific application or to asses the expected performance improvement from proposed improvements in subsystem performance. Sensor subsystem performances are generally specified in terms of the RMS values of sensor error sources such as those described above.

In any case, the fundamental tool in error budgeting is the Riccati equation of the Kalman filter. This requires a Kalman filter model for these error sources, specified in terms of the fundamental parameters of the Kalman filter model. These are the matrices F (or Φ), Q, H, and R. Formulas for these parameters are derived in the following sections.

Furthermore, because the impact of INS error sources such as scale factor errors on navigation errors depends on the sensor inputs, the solution of the Riccati equation requires simulating representative dynamic conditions for the intended INS applications.

10.4.5 INS Navigation Error Dynamics

Although it is simple in concept, inertial navigation has rather complex error propagation models. Be aware that INS navigation error analysis can get fairly complicated. Perhaps the only book on the subject currently in print is the one by Britting [12], written about half a century ago. More recent derivations for the inertial navigation error propagation models presented below can be found in Chapter 11 of Reference 2. These are models for navigation on or near the surface of Earth, although they would apply as well—with appropriate changes of parameter values—to any near-spherical or ellipsoidal planet with a well-defined surface.

10.4.5.1 Navigation Error Coordinates

ENU Coordinates The dominant error mechanisms for navigating in the near-earth environment depend on the direction of the (unmeasured) gravitational acceleration (downward) and the direction of the Earth rotation axis (polar). These have fixed directions with respect to "locally level" coordinates, either NED or ENU. Navigators of vehicles with well-defined roll, pitch, and yaw coordinates (Figure 10.19) may prefer roll–pitch–yaw (RYP) coordinates, which align with NED coordinates when the vehicle is right-side-up and headed North. Navigators who feel more comfortable with the vertical axis in the direction of increasing altitude and the others in the directions of increasing longitude and latitude, on the other hand, might prefer

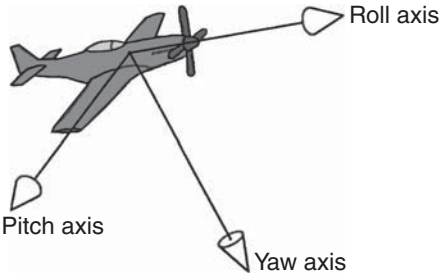


Figure 10.19 Roll–pitch–yaw coordinates.

ENU coordinates. Both GNSS and inertial navigation use altitude for the near-Earth environment, so derivations here will use ENU coordinates.

Sensor Coordinates For strapdown systems, the sensor input axes will have fixed directions with respect to vehicle-fixed RPY coordinates. An integral part of the INS installation and alignment process is the determination of the coordinate transformation matrix from sensor-fixed to vehicle-fixed coordinates,

$$C_{\mathrm{RPY}}^{\mathrm{SEN}}.$$

An integral part of the strapdown inertial navigation solution is the determination of the coordinate transformation matrix from sensor-fixed coordinates to navigation coordinates,

$$C_{\mathrm{NAV}}^{\mathrm{SEN}},$$

which is also used in the transformation of gimbaled inertial navigation error models to equivalent strapdown inertial navigation error models.

Vehicle attitude in navigation coordinates is then characterized by the coordinate transformation matrix

$$C_{\mathrm{RPY}}^{\mathrm{NAV}} = C_{\mathrm{RPY}}^{\mathrm{SEN}} \left(C_{\mathrm{NAV}}^{\mathrm{SEN}} \right)^{\mathrm{T}} \tag{10.81}$$

from navigation to vehicle-fixed coordinates.

10.4.5.2 First-Order Dynamic Modeling Inertial navigation errors are generally so small compared to the navigation solution that first-order approximations can be used in modeling their dynamics during navigation. However, this assumption may not hold for some methods of INS initialization using auxiliary sensors, for which initial orientation uncertainties can be much larger.

Because an INS is self-contained, errors in one part of the navigation solution easily couple into other parts of the solution. This is illustrated by the process flow diagram in Figure 10.20, in which some boxes are numbered according to the error coupling mechanisms they represent:

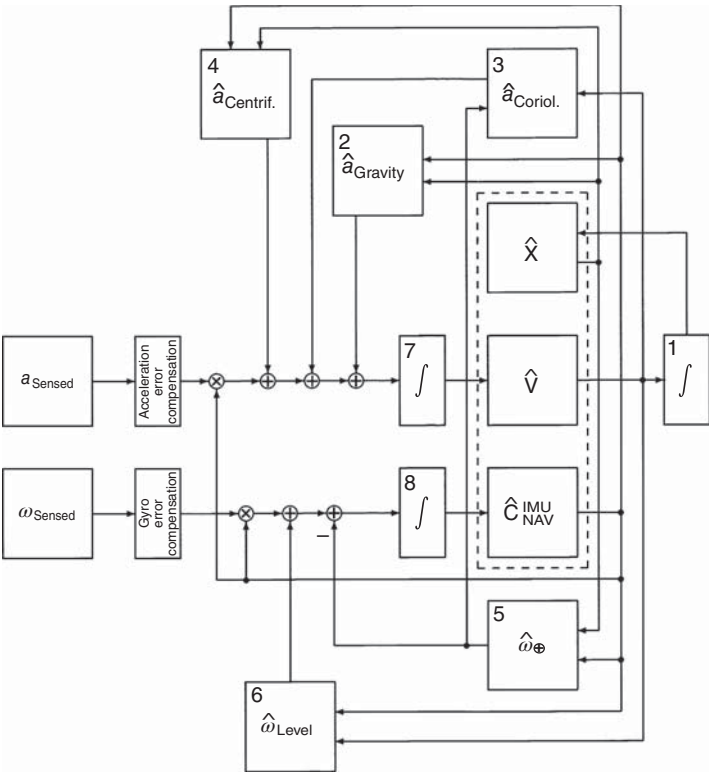


Figure 10.20 INS processing data flows.

1. Velocity errors couple into position errors through integration.
2. Position and attitude errors couple into acceleration errors through gravity compensation.
3. Velocity and Earthrate modeling errors couple into attitude rate errors through Coriolis compensation.
4. Position and attitude errors couple into acceleration errors through centrifugal compensation.
5. Position and attitude errors couple into attitude rate errors through Earthrate compensation.
6. Velocity and attitude errors couple into attitude rate errors through leveling.
7. Acceleration errors couple into velocity errors through integration.
8. Attitude rate errors couple into attitude errors through integration.
9. Acceleration errors couple into velocity errors through integration.
10. Attitude rate errors couple into attitude errors through integration.

For a more detailed treatment of INS error modeling, see References 12, 13, and 35 or Chapter 11 of Reference 2.

10.4.5.3 Nine-State Dynamic Model The nine state variables of inertial navigation error are the three components of position error, three components of velocity error, and three components of attitude error. In locally level "ENU" coordinates, these will be

ε_E, INS easting error (m)
ε_N, INS northing error (m)
ε_U, INS altitude error (m)
$\dot{\varepsilon}_E$, INS east velocity error (m/s)
$\dot{\varepsilon}_N$, INS north velocity error (m/s)
$\dot{\varepsilon}_U$, INS vertical velocity error (m/s)
ρ_E, INS misalignment about east axis (rad)
ρ_N, INS misalignment about north axis (rad)
ρ_U, INS misalignment about vertical axis (rad),

which can be represented as the components of an error dynamic model state vector

$$\varepsilon_{\text{NAV}} \stackrel{\text{def}}{=} \begin{bmatrix} \varepsilon_E & \varepsilon_N & \varepsilon_U & \dot{\varepsilon}_E & \dot{\varepsilon}_N & \dot{\varepsilon}_U & \rho_E & \rho_N & \rho_U \end{bmatrix}^{\text{T}}. \tag{10.82}$$

The resulting 9×9 dynamic coefficient matrix for the nine INS error variables can then be partitioned into block form with 3×3 blocks as

$$\frac{d\varepsilon_{\text{NAV}}(t)}{dt} = \mathbf{F}(t)\varepsilon_{\text{NAV}}(t) + w(t) \tag{10.83}$$

$$\mathbf{F} = \begin{bmatrix} 0_{3\times3} & I_3 & 0_{3\times3} \\ F_{21} & F_{22} & F_{23} \\ F_{21} & F_{22} & F_{23} \end{bmatrix} \tag{10.84}$$

$$F_{21} = \begin{bmatrix} 0 & 2\dfrac{\Omega_\oplus \sin\ \phi v_U}{\overline{R}_\oplus} + 2\dfrac{\Omega_\oplus \cos\ \phi v_N}{\overline{R}_\oplus} & 0 \\ 0 & -2\dfrac{\Omega_\oplus \cos\ \phi v_E}{\overline{R}_\oplus} + {\Omega_\oplus}^2 \sin^2\phi - {\Omega_\oplus}^2 \cos^2\phi & -{\Omega_\oplus}^2 \sin\ \phi \cos\ \phi \\ 0 & -2\dfrac{\Omega_\oplus \sin\ \phi v_E}{\overline{R}_\oplus} - 2{\Omega_\oplus}^2 \sin\ \phi \cos\ \phi & 2\dfrac{\text{GM}_\oplus}{{\overline{R}_\oplus}^3} + {\Omega_\oplus}^2 \cos^2\phi \end{bmatrix} \tag{10.85}$$

$$F_{22} = \begin{bmatrix} 0 & 2\Omega_\oplus \sin\ \phi & -2\Omega_\oplus \cos\ \phi \\ -2\Omega_\oplus \sin\ \phi & 0 & 0 \\ 2\Omega_\oplus \cos\ \phi & 0 & 0 \end{bmatrix} \tag{10.86}$$

$$F_{23} = \tag{10.87}$$

$$\begin{bmatrix} 2\Omega_\oplus \sin\ \phi v_U + 2\Omega_\oplus \cos\ \phi v_N \\ a_U + \dfrac{\mathrm{GM}_\oplus}{\overline{R}_\oplus{}^2} - 2\Omega_\oplus \cos\ \phi v_E - \Omega_\oplus{}^2 \cos\ {}^2\phi \overline{R}_\oplus \\ -a_N - 2\Omega_\oplus \sin\ \phi v_E - \Omega_\oplus{}^2 \sin\ \phi \cos\ \phi \overline{R}_\oplus \end{bmatrix}$$

$$\begin{bmatrix} -a_U - \dfrac{\mathrm{GM}_\oplus}{\overline{R}_\oplus{}^2} + \Omega_\oplus{}^2 \cos\ {}^2\phi \overline{R}_\oplus & a_N + \Omega_\oplus{}^2 \sin\ \phi \cos\ \phi \overline{R}_\oplus \\ 2\Omega_\oplus \sin\ \phi v_U & -a_E - 2\Omega_\oplus \cos\ \phi v_U \\ a_E - 2\Omega_\oplus \sin\ \phi v_N & 2\Omega_\oplus \cos\ \phi v_N \end{bmatrix}$$

$$F_{31} = \begin{bmatrix} 0 & 0 & \dfrac{v_N}{\overline{R}_\oplus{}^2} \\ 0 & \dfrac{\Omega_\oplus \sin\ \phi}{\overline{R}_\oplus} & -\dfrac{v_E}{\overline{R}_\oplus{}^2} \\ 0 & -\dfrac{\Omega_\oplus \cos\ \phi}{\overline{R}_\oplus} & 0 \end{bmatrix} \tag{10.88}$$

$$F_{32} = \begin{bmatrix} 0 & -\overline{R}_\oplus{}^{-1} & 0 \\ \overline{R}_\oplus{}^{-1} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{10.89}$$

$$F_{33} = \begin{bmatrix} -\dfrac{v_U}{\overline{R}_\oplus} & -\Omega_\oplus \sin\ \phi & \Omega_\oplus \cos\ \phi \\ \Omega_\oplus \sin\ \phi & -\dfrac{v_U}{\overline{R}_\oplus} & 0 \\ -\Omega_\oplus \cos\ \phi + \dfrac{v_E}{\overline{R}_\oplus} & \dfrac{v_N}{\overline{R}_\oplus} & 0 \end{bmatrix}, \tag{10.90}$$

where the INS error model dynamic coefficient matrix is defined in terms of the following parameters and variables:

$$\left.\begin{array}{lcll}
\overline{R}_{\oplus} & \stackrel{\text{def}}{=} & \text{mean Earth radius} & \approx\ 0.6371009 \times 10^{7}(\text{m}) \\
\text{GM}_{\oplus} & \stackrel{\text{def}}{=} & \text{Earth gravity constant} & \approx\ 0.3986004 \times 10^{15}(\text{m}^3/\text{s}^2) \\
\Omega_{\oplus} & \stackrel{\text{def}}{=} & \text{Earth rotation rate} & \approx\ 0.7292115 \times 10^{-4}(\text{rad/s}) \\
\widehat{\phi} & = & \phi + \varepsilon_N / \overline{R}_{\oplus} & \text{latitude (rad)} \\
\widehat{\theta} & = & \theta + \varepsilon_E / \left(\overline{R}_{\oplus} \cos\ \phi \right) & \text{longitude (rad)} \\
\hat{\text{E}} & = & E + \varepsilon_E & \text{easting with respect to INS (m)} \\
\widehat{N} & = & N + \varepsilon_N & \text{northing with respect to INS (m)} \\
\hat{h} & = & h + \varepsilon_U & \text{altitude (m)} \\
\widehat{v}_E & = & v_E + \dot{\varepsilon}_E & \text{east INS velocity (m/s)} \\
\widehat{v}_N & = & v_N + \dot{\varepsilon}_N & \text{north INS velocity (m/s)} \\
\widehat{v}_U & = & v_U + \dot{\varepsilon}_U & \text{vertical INS velocity (m/s).}
\end{array}\right\} \quad (10.91)$$

Here, the "hatted" ($\widehat{\ }$) variables are provided by the INS and updated by a Kalman filter in GNSS/INS integration.

These equations are implemented in the MATLAB function `Fcore9.m` on the accompanying Wiley web site. More detailed derivations can be found in Chapter 11 of Reference 2.

10.4.5.4 Universality of the Core Model The nine-state model for inertial navigation errors is fundamental to all types of INS implementations—gimbaled and strapdown. The only differences between gimbaled and strapdown models will be a factor of the transformation matrix

$$C_{\text{NAV}}^{\text{SEN}}$$

defined in Section 10.4.5.1. The same applies for carouseled or indexed gimbal implementations.

10.4.5.5 Seven-State Dynamic Model The seven-state model is for an INS in which altitude and altitude rate are not part of the INS solution but are determined independently from other sources—such as altimeter outputs. The error model omits the vertical components of position (altitude) and velocity (altitude rate) in the nine-sate model. This avoids the problem of vertical channel instability, and this model is commonly used for predicting CEP rates .

Formulas for the resulting 7×7 dynamic coefficient matrix are implemented in the MATLAB function `Fcore7.m` on the accompanying Wiley web site.

10.4.5.6 Common Characteristics of INS Errors When inertial navigation technology was being developed in the United States after WWII, much of the practical and theoretical error modeling had already been done by German engineers and scientists working on aircraft and missile inertial instrumentation before and during the War.

Schuler Oscillations The coupling of position errors into gravity compensation errors in the INS implementation creates an oscillation of position errors with the same period as a satellite at the same altitude as the INS, called the *Schuler period.* It is named after Maximilian Schuler (1882–1972), who discovered this phenomenon in his 1923 analysis [14] of a gyrocompass designed by his cousin Hermann Anschütz-Kaempfe (1832–1931). The gyrocompass design uses a gyroscope held in a fixed orientation with respect to local vertical by a pendulum linkage. Schuler was able to show that the effective pendulum period of the supporting linkage would have to be around 84.4 min to keep lateral accelerations from interfering with its operation. When the same oscillation period was found in inertial navigation errors, they came to be called *Schuler oscillations*.

INS implementations are usually tested in the laboratory by initializing with velocity errors and looking for the characteristic Schuler oscillation that should result. The same test also serves to verify INS simulation software. Figure 10.21 is a plot of simulated INS errors generated using the MATLAB INS Toolbox from GPSoft [15]. This shows how an initial northward velocity error of 0.1 m/s excites Schuler oscillations in the INS navigation errors, and how Coriolis accelerations rotate the direction of oscillation—like a Foucault pendulum with Schuler period. The total simulated time is about 14 h, time enough for 10 Schuler oscillation periods.

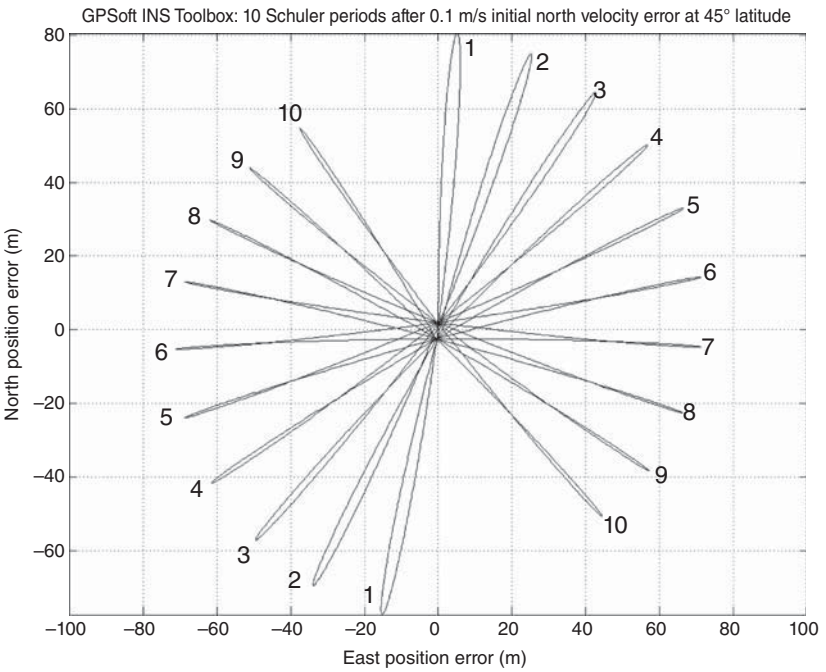


Figure 10.21 GPSoft INS Toolbox simulation of Schuler oscillations.

Vertical Channel Instability This is another INS error vulnerability that had been discovered before inertial navigation was reduced to practice. In 1946, when serious technical development of nuclear weapon delivery systems was just getting under way in the United States, Russian-born American scientist George Gamow (1904–1968) was part of a government technology advisory committee. He advised against military support for development of inertial navigation on the grounds that it was inherently unstable in the vertical direction. Gamow was correct, but he was eventually outmaneuvered by Charles Stark Draper, founder of the MIT Instrumentation Laboratory and considered to be the "father" of inertial navigation. Draper would demonstrate that the problem could be solved for aircraft navigation by aiding with a barometric altimeter, already standard equipment aboard most aircraft. This was perhaps the first time additional sensors were used for aiding INS navigation.

It is the theoretical gravity gradient model that causes the instability. In essence, having a positive altitude error causes an error in the calculated downward gravitational acceleration, which leads to exponential growth of the altitude error with an exponential time constant in the order of several minutes at sea level.

Performance Degradation over Time Inertial navigation without aiding (except in the vertical channel) is called *free-inertial* navigation. Other than horizontal Schuler oscillation and vertical channel instability, most free-inertial navigation errors do tend to degrade with time. This is not terribly serious for ballistic missile guidance, in which the total INS navigation/guidance phase lasts only for the first few minutes of launch. It is much more serious for "cruise" applications such as for navigation of aircraft and watercraft operating on or near Earth's surface.

CEP Rate For cruise applications, free-inertial navigation performance classes have been defined in terms of the rate at which navigation errors grow with time. The established measure of performance has been the rate of growth of the CEP (defined in Section 10.2.5.2). The unit of distance used is the *nautical mile* (defined in Section 10.2.5.1)), and the rate of growth is specified in *nautical miles per hour.*

INS Performance Classes In the 1970s the US Air Force defined three classes of INS performance:

High accuracy INS navigation accuracy degrades at 0.1 NMi/h CEP rate. This originally included systems designed for strategic military operations, although most of these systems eventually surpassed the specification.

Medium accuracy INS navigation accuracy degrades at 1 NMi/h CEP rate. This applied to most aircraft, including commercial aircraft—before GPS.

Low accuracy INS navigation accuracy degrades at 10 NMi/h CEP rate. This applied to short-range tactical missiles, for example.

All this would change when satellite navigation became available.

10.4.6 INS Sensor Compensation Parameter Errors

10.4.6.1 Inertial Sensor Calibration When the Kalman filter was introduced in the 1960s, it was soon put to use in determining parameters of the formulas used for compensating the sensors used in inertial navigation. It was already known that gimbaled systems could be calibrated in the laboratory (and in missile silos) by using their gimbals to orient the ISA in certain directions with respect to the local vertical (to control input acceleration) and Earth rotation axis (to control input rotation rates). Covariance analysis using the Riccati equation could then quantify INS performance as a function of sensor error characteristics and calibration procedures. This approach was soon extended to include self-calibration using many types of external navigation aids.

10.4.6.2 Sensor Compensation Parameter Errors Stanley F. Schmidt, who might be considered to be the "father" of INS integration with other sensor systems, was instrumental in developing a navigation system integrating INS and airborne radar for the C5A aircraft [16] in the mid-1960s.[14]

Starting in 1946, the Northrop Corporation[15] developed a series of cruise missiles with gimbaled INS, some of them with star trackers mounted on the stable element. The earliest designs used the star tracker to servo the stable element to keep it horizontal and aligned to true north. Later designs—including the navigator for the U-2 spycraft—would use a Kalman filter to correct sensor errors, as well.

Some of the work done in secrecy in the 1960s and 1970s would determine which types of auxiliary sensors could be used for keeping INS navigation errors in check by recalibrating the INS sensors on the fly. With the availability of a civilian channel on the GPS navigation satellites in the early 1990s, much of this technology would migrate to the public domain.

10.4.6.3 INS Sensor Calibration and Compensation An integral part of GNSS/INS integration is the ability to estimate and correct drifting values of the parameters used in compensating for sensor errors in inertial navigation. This capability reduces sensor stability requirements and costs. The following is a brief tutorial, including derivations of the models used for that purpose.

Sensor Calibration Because manufacturing tolerances on inertial sensors could not be made tight enough for the demands of navigation, it soon became standard practice to get the last few bits of accuracy by compensating the outputs of the sensors for any known residual error characteristics. This required a parametric model for the expected errors as a function of the sensor outputs and a calibration procedure in

[14]Most military applications of Kalman filtering during this era were classified, but this may have been the first use of the Kalman filter and other sensors to correct INS instrument errors during navigation. The integration of the Navy Transit navigation satellite system with ship-board INS systems also occurred in this time frame, but technical details are scant.

[15]The third corporation founded by aerospace technology pioneer Jack Northrop (1895–1981) and bearing his name.

which the outputs were measured with known inputs. A Kalman filter could then be used to estimate the parameters of the model, given the input/output data.

Affine Compensation Parameters Among the more common sensor errors requiring calibration and compensation in this way were terms such as

- Sensor bias, which equals the sensor output when there is zero sensor input.
- Sensor scale factor, which is the first-order variation of output with input.
- Sensor input axis misalignment errors.

These, it turns out, can all be modeled as an affine transformation at the level of three, nominally orthogonal sensors (gyroscopes or accelerometers) as

$$z_{\text{output}} = Mz_{\text{input}} + b, \tag{10.92}$$

where M is a 3×3 matrix and b is a column 3-vector. The values of M and b can be determined by least-squares or Kalman filter fitting of input–output pairs. Once M and b have been determined, the affine transformation is easily inverted to obtain the *error compensation formula* as an affine transformation:

$$z_{\text{input}} = Nz_{\text{output}} + d \tag{10.93}$$

$$N = M^{-1} \tag{10.94}$$

$$d = -Nb, \tag{10.95}$$

or the compensation model parameters N and d can be determined directly by least-squares fitting from the same set of input–output pairs.

Affine transformations are just the zeroth-order (bias) and first-order (scale factor and misalignment) terms in the power series expansion of the input/output relationship, and this same modeling and compensation procedure can be extended to any order. For many sensor designs and applications, however, the zeroth-order and first-order terms are the dominant errors.

There may also be other sensor output errors modeled as sensitivities of gyroscopes to acceleration, accelerometers to rotation rates, or both to temperature. These, too are generally represented by parametric models, the parameters of which can be calibrated under controlled conditions.

10.4.6.4 Drifting Compensation Parameters

Parameter Drift Inertial sensors are subject to all sorts of errors, many of which can be modeled, calibrated, and compensated. Sensor failure modes may include abrupt changes in input/output characteristics due to subcomponent failures. Some high reliability INS designs have included redundant sensors, and software to detect suspected failures of this sort and switch to using the remaining sensors. Beyond that, there is a class of slowly creeping input/output characteristics attributed to "compensation parameter drift."

Causes and Remedies Many sensor error compensation parameters (especially those of the more expensive sensors) have relatively stable values, but some may tend to drift slowly but unpredictably over time. This drifting may be due to a number of factors, including ambient temperature variations. Temperature sensitivities can be compensated to some degree by calibrating the effect as a function of sensor temperature and compensating for the error by using one or more temperature sensors on the ISA during operation. But some drift phenomena are not so easily understood and modeled, and are often simply attributed to "aging" of sensors. This may be due, in part, to slow creep of materials due to internal stresses from manufacturing processes, but such processes are generally not sufficiently predictable to be modeled and compensated for. Among those sensor error compensation parameters commonly subject to such "aging" are biases and scale factors. In the next section, we will use scale factor and bias parameters to demonstrate how GNSS/INS integration can actually correct for drift in the compensation model parameter values. For that purpose, we will need to model these parameters in terms of how they couple into navigation solution error.

Drift Dynamic Modeling The best models come from analysis of the actual drift data from sensors under operating conditions. It would be useful, for example, to know whether the actual drift resembles a truly constant parameter, an exponentially correlated process, or a random walk, and to know in the latter cases what is the approximate driving noise covariance and correlation time. For example, for some sensor technologies, input axis directions tend to be relatively stable compared to biases and scale factors (especially for sensors with limited temperature control).

Random Walk Models If we assume random walk models for just the biases and scale factors, for example, the dynamic models for the compensation parameter errors will have the form

$$\varepsilon_{ca} \overset{\text{def}}{=} \begin{bmatrix} \varepsilon_{saE} \\ \varepsilon_{saN} \\ \varepsilon_{saU} \\ \varepsilon_{baE} \\ \varepsilon_{baN} \\ \varepsilon_{baU} \end{bmatrix} \tag{10.96}$$

$$\frac{d}{dt}\varepsilon_{ca} = w_{ca}(t) \tag{10.97}$$

$$\underset{W_{ca}}{\mathrm{E}}\,\langle w_{ca}(t)\rangle = 0 \tag{10.98}$$

$$\underset{W_{ca}}{\mathrm{E}}\,\langle w_{ca} w_{ca}^{\mathrm{T}}\rangle = Q_{ca} \tag{10.99}$$

$$\varepsilon_{c\omega} \stackrel{\text{def}}{=} \begin{bmatrix} \varepsilon_{s\omega E} \\ \varepsilon_{s\omega N} \\ \varepsilon_{s\omega U} \\ \varepsilon_{b\omega E} \\ \varepsilon_{b\omega N} \\ \varepsilon_{b\omega U} \end{bmatrix} \tag{10.100}$$

$$\frac{d}{dt}\varepsilon_{c\omega} = w_{c\omega}(t) \tag{10.101}$$

$$\mathop{\mathrm{E}}_{W_{c\omega}} \langle w_{c\omega}(t) \rangle = 0 \tag{10.102}$$

$$\mathop{\mathrm{E}}_{W_{c\omega}} \langle w_{c\omega} w_{c\omega}^{\mathrm{T}} \rangle = Q_{c\omega}, \tag{10.103}$$

(10.104)

where the subscripts

$_{ca}$ refers to accelerometer compensation parameter errors,
$_{sa}$ refers to accelerometer scale factor errors,
$_{ba}$ refers to accelerometer bias errors,
$_{c\omega}$ refers to rate gyro compensation parameter errors,
$_{s\omega}$ refers to rate gyro scale factor errors,
$_{b\omega}$ refers to rate gyro bias errors,

and the subscript post indices $_{E}$, $_{N}$, and $_{U}$ refer to the individual input axes. This choice would imply an ENU gimbaled system, but the same sort of indexing can be done using host/carrier-fixed axes for a strapdown system, or similar axes for carouseled systems.

Determining Model Parameters The covariance matrices of dynamic disturbance noise Q_{ca} and $Q_{c\omega}$ will be diagonal if the random walk processes are independent for the different sensor compensation parameters. However, because performance is generally improved if the model parameters are close to correct, it is usually worth verifying independence by analyzing the actual drift histories. Intermediate integrated GNSS/INS navigation results can provide the drift histories for such analysis.

10.4.6.5 Compensation Error Coupling into Navigation Errors Figure 10.22 illustrates how errors in scale factor and bias parameters create navigation errors. Figure 10.22(a) is for accelerometers, and Figure 10.22(b) is for gyroscopes. The resulting errors in acceleration and attitude rate would then be represented in algebraic form as

$$\varepsilon_a = \underbrace{\begin{bmatrix} D_a & I_3 \end{bmatrix}}_{F_{24}} \begin{bmatrix} \varepsilon_{sa} \\ \varepsilon_{ba} \end{bmatrix} \tag{10.105}$$

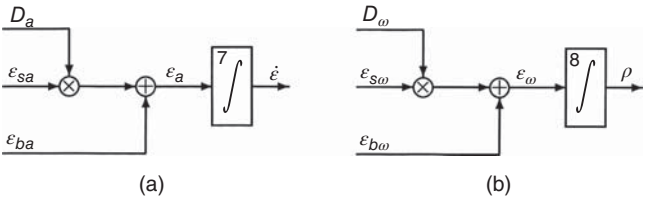


Figure 10.22 Sensor scale factor and bias compensation error flows.

$$D_a = \text{diag}[a_E,\ a_N,\ a_U] \tag{10.106}$$

$$\varepsilon_\omega = \underbrace{\begin{bmatrix} D_\omega & I_3 \end{bmatrix}}_{F_{35}} \begin{bmatrix} \varepsilon_{s\omega} \\ \varepsilon_{b\omega} \end{bmatrix} \tag{10.107}$$

$$D_\omega = \text{diag}[\omega_E,\ \omega_N,\ \omega_U], \tag{10.108}$$

where the significance of F_{24} and F_{35} is explained below.

Accelerometer compensation errors couple into acceleration errors, which would be input to the box numbered "7" in Figure 10.20. Rate gyro compensation errors couple into attitude rate errors, which would be input to the box numbered "8" in Figure 10.20.

10.4.6.6 Augmented Dynamic Coefficient Matrix Adding just the 12 state variables for bias and scale factor drift on each of the six sensors (three for acceleration and three for attitude rate) to the nine core navigation error state variables would yield a 21×21 augmented dynamic coefficient matrix:

$$F_{\text{aug}} = \begin{bmatrix} 0_{3\times3} & I_3 & 0_{3\times3} & 0_{3\times6} & 0_{3\times6} \\ F_{21} & F_{22} & F_{23} & F_{24} & 0_{3\times6} \\ F_{31} & F_{32} & F_{33} & 0_{3\times6} & F_{35} \\ 0_{6\times3} & 0_{6\times3} & 0_{6\times3} & 0_{6\times6} & 0_{6\times6} \\ 0_{6\times3} & 0_{6\times3} & 0_{6\times3} & 0_{6\times6} & 0_{6\times6} \end{bmatrix} \tag{10.109}$$

$$F_{24} = \begin{bmatrix} a_E & 0 & 0 & 1 & 0 & 0 \\ 0 & a_N & 0 & 0 & 1 & 0 \\ 0 & 0 & a_U & 0 & 0 & 1 \end{bmatrix} \tag{10.110}$$

$$F_{35} = \begin{bmatrix} \omega_E & 0 & 0 & 1 & 0 & 0 \\ 0 & \omega_N & 0 & 0 & 1 & 0 \\ 0 & 0 & \omega_U & 0 & 0 & 1 \end{bmatrix} \tag{10.111}$$

for the random walk parameter drift model. As we have mentioned before, the acceleration and attitude rate vector components must be in sensor input-axis coordinates for this to work. The "ENU" coordinates are only correct for a locally level ENU gimbaled implementation. Otherwise, the input accelerations and attitude rates must be

transformed from navigation coordinates to sensor input axis coordinates, then back to navigation coordinates. In practice, the outputs of the sensors would work for the first part, but the resulting acceleration and rotation rate errors need to be transformed back to navigation coordinates (which, in this case, are locally level ENU).

10.4.6.7 Strapdown and Carouseled Model Differences The above derivations of compensation drift error models are for gimbaled systems in which ISA-fixed coordinates are east, north, and up, in which case the inputs of the sensors are in navigation (NAV) coordinates.

For carouseled or α-wander gimbaled implementations, the ISA axes are rotated through a known angle α about the vertical axis, in which case the ISA-to-NAV coordinate transformation matrix is

$$C_{\text{ISA}\rightarrow\text{NAV}} = \begin{bmatrix} \cos(\alpha) & \sin(\alpha) & 0 \\ -\sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{10.112}$$

In that case, the only difference will be in the definitions of the submatrices F_{24} (Equation 10.109) and F_{35} (Equation 10.110) of the system error dynamic coefficient matrix as

$$F_{24} = C_{\text{ISA}\rightarrow\text{NAV}} \begin{bmatrix} a_E & 0 & 0 & 1 & 0 & 0 \\ 0 & a_N & 0 & 0 & 1 & 0 \\ 0 & 0 & a_U & 0 & 0 & 1 \end{bmatrix} \tag{10.113}$$

$$F_{35} = C_{\text{ISA}\rightarrow\text{NAV}} \begin{bmatrix} \omega_E & 0 & 0 & 1 & 0 & 0 \\ 0 & \omega_N & 0 & 0 & 1 & 0 \\ 0 & 0 & \omega_U & 0 & 0 & 1 \end{bmatrix}, \tag{10.114}$$

where the acceleration a in D_a and ω in D_ω are now in sensor-fixed ISA coordinates, which can be implemented using the compensated outputs of the accelerometers (in ISA coordinates) before their transformation to NAV coordinates. The required value of $C_{\text{ISA}\rightarrow\text{NAV}}$ is already part of the navigation solution.

For strapdown implementations, the sensor-fixed inputs are also in sensor-fixed (ISA) coordinates, which are no longer the NAV coordinates. But the ISA-to-NAV coordinate transformation matrix $C_{\text{ISA}\rightarrow\text{NAV}}$ is still part of the navigation solution. In this case, that will be the value of $C_{\text{ISA}\rightarrow\text{NAV}}$ to be used in Equations 10.112 and 10.113.

10.4.7 MATLAB Implementations

10.4.7.1 Initial Navigation Uncertainties Initial navigation errors do have a significant impact of the ensuing CEP rates determined during testing. Standard INS performance testing typically includes the approved initial alignment and navigation initialization for the system under test, which we have not defined here.

As an alternative, we have assumed a standard, relatively accurate, initial navigation solution in all cases. The RMS uncertainties for this initialization error budget are listed in Table 10.4.

TABLE 10.4 RMS Initial Navigation Uncertainties

Navigation Variable	RMS Value	Units
Easting	2	(m)
Northing	2	(m)
Altitude	2	(m)
East Velocity	10^{-1}	(m/s)
North Velocity	10^{-1}	(m/s)
Altitude Rate	10^{-1}	(m/s)
East Tilt	10^{-8}	(rad)
North Tilt	10^{-8}	(rad)
Heading	10^{-8}	(rad)

These have been made quite small so that the resulting estimates of CEP rates are influenced more by INS error budgets and less by initialization errors. That luxury is generally not allowed in practice, however.

10.4.7.2 Taming Vertical Channel Instability This implementation uses a barometric altimeter to stabilize the altitude of an INS otherwise operating in "free-inertial" mode. That is, the INS navigates in 3D but uses altimeter measurements to estimate and correct its vertical position and velocity errors. The barometer is assumed to be corrected for local air pressure variations by using another barometer at a fixed location to telemeter the local correction. The resulting RMS altitude bias error is assumed to be 2 m with a correlation time constant of 1 h, plus RMS white noise of 0.1 m. The INS is assumed to be of "medium accuracy," having a CEP rate in this augmented mode of 1 NMi/h—as shown in Figure 10.23.

The full 22-state error model (9 navigation errors, plus 12 sensor compensation errors, plus 1 altimeter bias error) used in the Riccati equation for assessing performance with and without the barometer is listed in the m-file `MediumAccuracyINS.m` on the Wiley web site. This evaluates RMS vertical channel errors during a simulated test on a 100-km figure-8 track, the results of which are plotted in Figure 10.24. This shows the resulting RMS altitude uncertainty over 4 h, with and without barometer aiding. The aided altitude uncertainty settles at around 2 m, whereas the RMS uncertainty for the unaided case climbs to more than 10^4 km in 4 h. The horizontal error also grows unreasonably without vertical channel stabilization.

With vertical channel stabilization, the computed CEP grows as shown in Figure 10.23, a least-squares straight-line fit to which yields an estimated CEP rate of 0.96 NMi/h for this particular simulated test trajectory with the assumed initial navigation uncertainties. This plot also exhibits a significant Schuler frequency component, which one might expect from initial navigation errors.

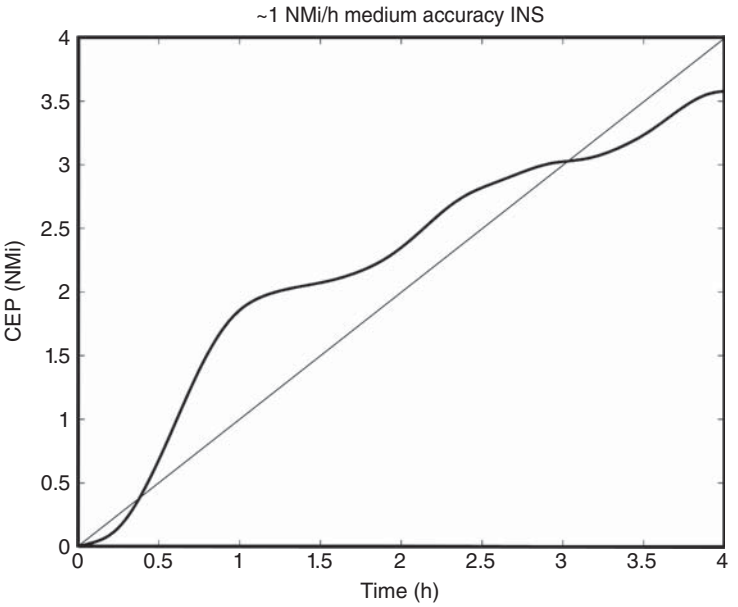


Figure 10.23 Medium accuracy INS: horizontal uncertainties versus time.

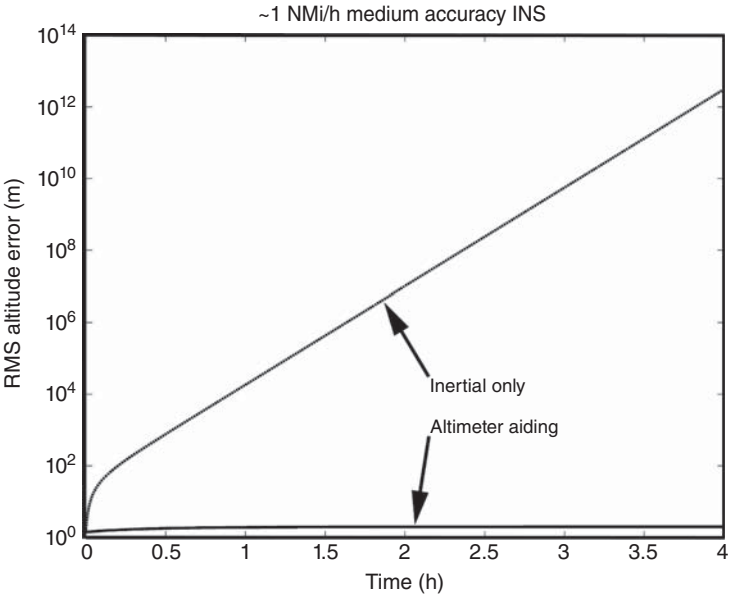


Figure 10.24 Medium accuracy INS: RMS altitude uncertainties versus time.

10.4.7.3 INS Error Budgeting Using Covariance Analysis Table 10.5 lists tentative error budgets[16] for what might be considered "high" (0.1 NMi/h CEP rate), "medium" (1 NMi/h CEP rate), and "low" accuracy (10 NMi/h CEP rate), inertial navigators, along with performance specifications for the barometric altimeter system used for stabilizing vertical navigation. The middle error budget has been used for demonstrating the efficacy of this form of vertical channel stabilization, shown in Figure 10.24.

The three m-files

`INSErrBud0point1NMiPerHr.m`	($\approx$ 0.1 NMi/h CEP Rate)
`INSErrBud1NMiPerHr.m`	($\approx$ 1 NMi/h CEP Rate)
`INSErrBud10NMiPerHr.m`	($\approx$ 10 NMi/h CEP Rate)

on the Wiley web site are designed to demonstrate how varying individual parameters in the corresponding INS error budgets will influence the resulting INS CEP rate in a specific dynamic simulation.

The common dynamic simulation is a 4-h run at 100 kph on a 100-km figure-8 test track, using a barometric altimeter for stabilizing vertical navigation.

Outputs include plots of CEP versus time and results of scaling individual error budget terms up and down by an order of magnitude. These error budgets include seven that are specific to the INS sensors, three related to the barometric altimeter used for altitude stabilization, and one (the intersample time period) related to simulation conditions. The standard values used in the m-files are listed in Table 10.5 for the three error budgets. The standard files used for saving the respective error budget as mat-files are also listed at the top of the table. These files can then be downloaded for use by other m-files.

The analysis in this case consists of estimating the resulting up- or down-scaling of the CEP rate when each INS error budget term is scaled up and down by an order of magnitude. Only the first seven error budget values are varied in the analysis, because the other values are not related to the INS. The eighth is the intersample interval and the rest are related to the auxilliary barometric altimeter used for vertical channel stabilization. Gyro-related parameters are specified in degrees per hour in the table but converted to radians per second internally for the simulations.

Example 10.3 (Error Budget Analysis for Medium Accuracy INS) This example uses the medium accuracy error budget from `INSErrBud1NMiPerHr.m`, which yields the CEP history shown in Figure 10.23 and the error budget analysis shown in Figure 10.25, which is a semi-log plot of the relative scaling effects on CEP as a function of parameter number.

[16]These error budgets are not based on actual INSs. They are only hypothetical examples which just happen to have the approximate CEP rates of high, medium, and low accuracy performance on the 100-km figure-8 test track simulator.

TABLE 10.5 Baseline INS Error Budgets for 0.1, 1, and 10 NMi/h CEP Rates

Parameter Sequence No.	Parameter Description	Units	CEP Rate 0.1 `INSEBHiAcc.mat`	CEP Rate 1 `INSEBMedAcc.mat`	CEP Rate 10 `INSEBLoAcc.mat`
1	RMS acceleration noise	m/s/sqrt(s)	10^{-6}	10^{-4}	10^{-4}
2	RMS gyro noise	deg/hr/sqrt(s)	10^{-4}	10^{-2}	10^{-2}
3	Sensor compensation error correlation time	s	15	600	1800
4	RMS Accel. Bias Err.	m/s/s	9.8×10^{-5}	9.8×10^{-5}	9.8×10^{-4}
5	RMS acceleration scale factor error	part/part	10^{-5}	10^{-5}	10^{-4}
6	RMS gyro bias error	deg/hr	10^{-4}	0.03	0.19
7	RMS gyro scale factor error	part/part	10^{-4}	10^{-4}	10^{-3}
8	intersample interval	s	1	1	1
9	RMS altimeter error	m	2	2	2
10	altimeter bias error correlation time	s	3600	3600	3600
11	RMS altimeter noise	m/sqrt(s)	0.1	0.1	0.1

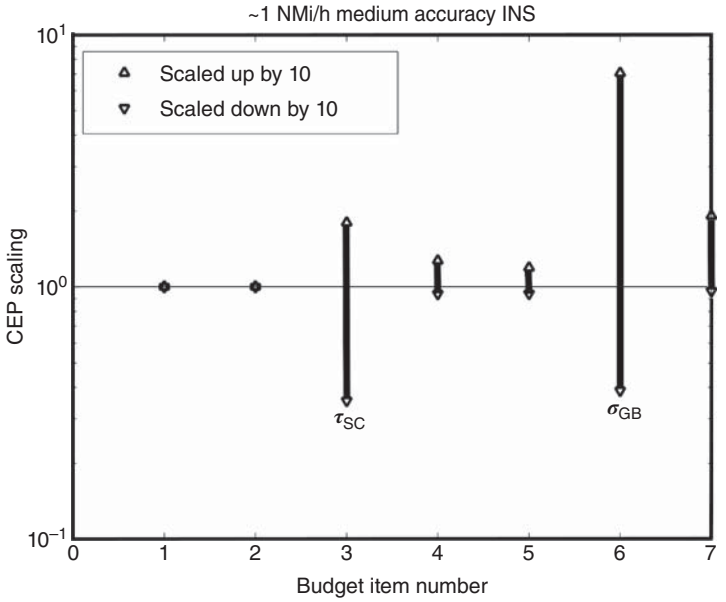


Figure 10.25 Relative CEP scaling from varying INS error budget parameters.

Varying the first two INS error budget parameters,

1. RMS Accelerometer Noise
2. RMS Gyro Noise

up and down by an order of magnitude has essentially no impact on CEP rate. This would indicate that these are already so small that they have little impact on CEP rate. From this, one might surmise that they can be derated by an order of magnitude or more with little or no impact on performance.

Varying the third parameter (sensor compensation error correlation time, τ_{SC}) up or down by a factor of 10 has a significant impact on CEP rate in both directions. Error correlation times are not that easy to change, however. Variation of performance with correlation time is not necessarily monotonic, either. Figure 10.26 is a plot of CEP rate as a function of τ_{SC} over several orders of magnitude. It shows that there are two values of τ_{SC} at which the resulting CEP rate is 1 NMi/h, with a peak in between. This type of dependence of behavior on correlation time is not that unusual.

The next two INS error budget parameters,

1. RMS accelerometer bias error
2. RMS accelerometer scale factor error,

show more downside performance degradation from loosening their values, but little upside gain from tightening them.

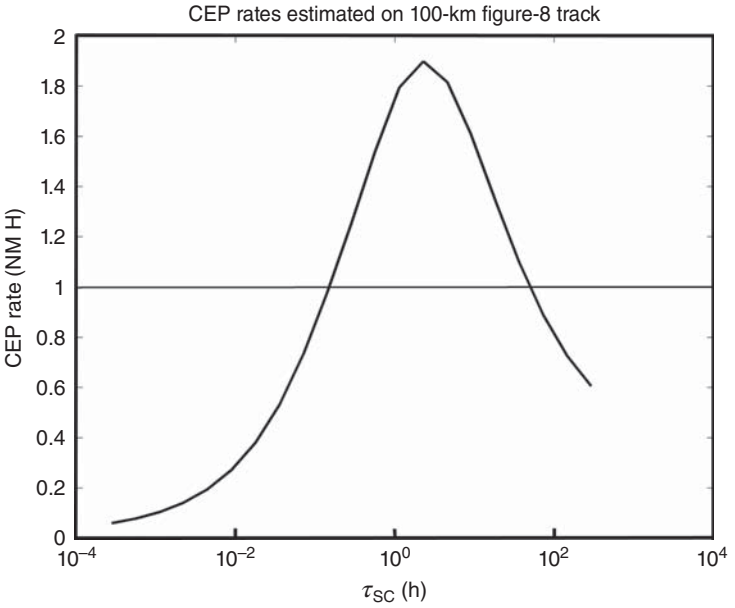


Figure 10.26 CEP rate versus t_{SC}.

The sixth parameter (RMS gyro bias error, σ_{GB}) shows about the same relative gain from tightening the requirement as τ_{SC}, and considerably more downside performance degradation from loosening it. This is perhaps the dominating error source, and the one to which one would look first to improve overall performance.

The final parameter, RMS gyro scale factor error, shows more downside gain from loosening it than upside gain from tightening it.

Error budget sensitivities for the higher and lower accuracy INS systems can be studied by running `INSErrBud0point1NMiPerHr.m` and `INSErrBud10NMiPerHr.m`, respectively.

These sensitivities of performance to error budget variations provide useful insight to the system designer, in terms of

1. What are the critical sensor performance parameters, in terms of their impact on expected INS CEP rate?
2. Where can sensor performance requirements be relaxed without much impact on navigation performance?
3. Where can tightening sensor performance requirements make the greatest improvement in navigation performance?
4. How can system cost be lowered without sacrificing performance?
5. How can performance be improved with the least cost?

These are always critical design considerations in the development of inertial sensors and systems.

10.5 GNSS/INS INTEGRATION

Stochastic models for INS and GNSS navigation errors were derived in the previous two sections. Integrated GNSS/INS navigation uses Kalman filters with models of those types plugged into the relevant parameter matrices F (or Φ), Q, H, and R. But we first need to establish some nomenclature used in GNSS/INS integration technologies.

10.5.1 Background

10.5.1.1 A Very Short History The earliest attempts at GNSS/INS integration were for the Navy Transit navigation satellite system used by Navy submarines in the 1960s, but much of this work had been cloaked in secrecy and lost to history. At around the same time, INS systems aboard military aircraft were being integrated with auxiliary sensors such as airborne radar, star trackers, LORAN, or Omega.

The earliest hardware demonstrations with Transit's successor began in the 1980s with GPS, before it was fully operational. In order to maintain lower technical and schedule risks, these began by using existing inertial navigators with minimal hardware modifications to demonstrate potential improvements in Air Force weapon delivery accuracies. It was less risky to proceed by stages, going from low technical complexity and cost to higher technical complexity and cost. Test results did confirm what had already been demonstrated in computer simulations with much more sophisticated implementation equations. Military applications of GPS/INS integration were already well established before there were opportunities for civilian applications.

10.5.1.2 Loosely and Tightly Coupled Implementations GNSS/INS integration architectures have been loosely labeled as "loosely coupled" or "tightly coupled" according to the degree to which the Kalman filter implementation alters the internal workings of each subsystem from its stand-alone configuration, as illustrated in Figure 10.27.

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. It may improve navigation accuracy while GNSS is available, but it defaults to free-inertial accuracy when it is not available.

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.

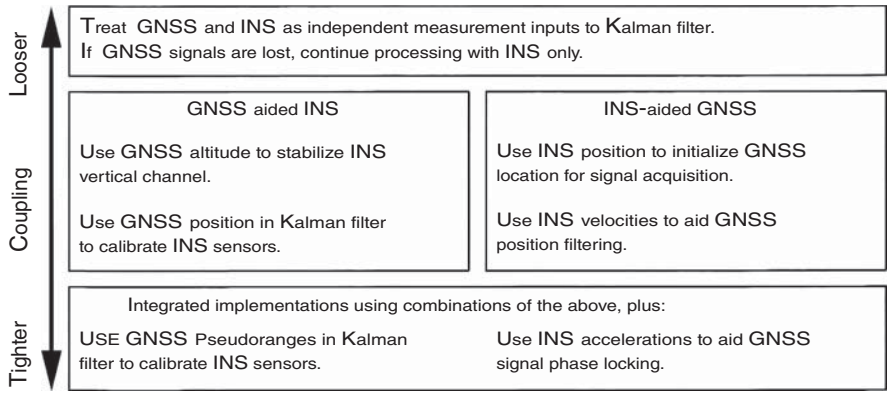


Figure 10.27 Loose/tight GSNN/INS integration strategies.

- 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 GNSS signal reacquisition is faster when signals do 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.

Another factor favoring more tightly coupled implementations is that their performance is generally less sensitive to the quality (and cost) of INS sensors. Historically, the INS had been 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 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.

10.5.1.3 Loosely Coupled Implementations These are mostly of historical interest today, because tightly coupled implementations offer the same benefits—and more.

1. *INS Position Initialization.* Medium and high accuracy inertial navigators are capable of sensing their latitude, but not their longitude. Manual entry of initial longitude is discouraged, because consequences of entry errors can be severe. GNSS-derived position is more reliable.

2. *INS Altitude Stabilization.* From the very beginning of inertial navigation, barometric altimeters have been used for stabilizing altitude error, as illustrated in Example 10.4.7.2. The function of the altimeter in this implementation can be replaced by a GNSS receiver, and the resulting performance can be modeled and analyzed in much the same way as for the altimeter. Both auxiliary sensors (altimeter or GNSS) have time-correlated errors, but GNSS errors tend to have shorter correlation times.
3. *INS-Aided GNSS Satellite Signal Phase Tracking.* GNSS receivers need to correct for satellite signal Doppler shifts, due to relatively high satellite velocities.[17] 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. 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. 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. In that case, attitude and attitude rate information from the INS can also aid the antenna-switching implementation.

Example 10.4 (INS Vertical Channel Damping with GNSS) We can model the dynamics of altitude error ε_h in inertial navigation systems as

$$\frac{d}{dt}\varepsilon_h = \dot{\varepsilon}_h \tag{10.115}$$

$$\frac{d}{dt}\dot{\varepsilon}_h = \frac{1}{\tau_S}\varepsilon_h + \varepsilon_a(t) \tag{10.116}$$

$$\frac{d}{dt}\varepsilon_a(t) = \frac{-1}{\tau_a}\varepsilon_a(t) + w(t) \tag{10.117}$$

$$\tau_S \approx 806.4 \text{ s}, \tag{10.118}$$

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

$$Q_{ta} = \frac{2\,\sigma_a^2}{\tau_a}, \tag{10.119}$$

where σ_a^2 is the mean-squared accelerometer error.

[17] Some few GNSS satellites may be in geosynchronous orbits. In order to maintain global coverage, however, most of the satellites will be in lower orbits with relatively high inclination angles to the equator.

The state-space model for the vertical channel in continuous time is then

$$x = \begin{bmatrix} \varepsilon_h \\ \dot{\varepsilon}_h \\ \varepsilon_a \end{bmatrix} \tag{10.120}$$

$$\dot{x} = \begin{bmatrix} 0 & 1 & 0 \\ \frac{1}{\tau_S} & 0 & 1 \\ 0 & 0 & \frac{-1}{\tau_a} \end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \\ w(t) \end{bmatrix} \tag{10.121}$$

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

$$\Phi = \exp\ (F\ \Delta t) \tag{10.122}$$

$$= \begin{bmatrix} \phi_{1,1} & \phi_{1,2} & \phi_{1,3} \\ \phi_{2,1} & \phi_{2,2} & \phi_{2,3} \\ 0 & 0 & \lambda_a \end{bmatrix} \tag{10.123}$$

$$\phi_{1,1} = \frac{\lambda_S^2 + 1}{2\ \lambda_S} \tag{10.124}$$

$$\phi_{1,2} = \frac{\tau_S(\lambda_S^2 - 1)}{2\ \lambda_S} \tag{10.125}$$

$$\phi_{1,3} = \frac{\tau_a {\tau_S}^2(\tau_S + \tau_a - 2\ \tau_a\lambda_a\lambda_S + \tau_a\lambda_S^2 - \tau_S\lambda_S^2)}{2\ \lambda_S(\tau_a^2 - \tau_S^2)} \tag{10.126}$$

$$\phi_{2,1} = \frac{(\lambda_S - 1)(\lambda_S + 1)}{2\ \tau_S\lambda_S} \tag{10.127}$$

$$\phi_{2,2} = \frac{\lambda_S^2 + 1}{\lambda_S} \tag{10.128}$$

$$\phi_{2,3} = \frac{\tau_S\tau_a(-\tau_S - \tau_a + 2\ \tau_S\lambda_a\lambda_S + \tau_a\lambda_S^2 - \tau_S\lambda_S^2)}{2\ \lambda_S(\tau_a^2 - \tau_S^2)} \tag{10.129}$$

$$\lambda_S = \exp\ (\Delta t/\tau_S) \tag{10.130}$$

$$\lambda_a = \exp\ (-\Delta t/\tau_a). \tag{10.131}$$

The corresponding process noise covariance in discrete time,

$$Q_a \approx \Delta t\ Q_{ta} \tag{10.132}$$

$$\approx \frac{2\ \Delta t\ \sigma_a^2}{\tau_a}. \tag{10.133}$$

The kth altitude outputs from the INS and GNSS receiver will be

$$\hat{h}_k^{[\mathrm{INS}]} = h_k^{[\mathrm{TRUE}]} + \varepsilon_h \tag{10.134}$$

$$\hat{h}_k^{[\mathrm{GNSS}]} = h_k^{[\mathrm{TRUE}]} + w_k, \tag{10.135}$$

respectively, where $h_k^{[\mathrm{TRUE}]}$ is the true altitude and w_k is the GNSS receiver altitude error.

The difference

$$z_k = \hat{h}_k^{[\mathrm{INS}]} - \hat{h}_k^{[\mathrm{GNSS}]} \tag{10.136}$$

$$= \varepsilon_h - w_k \tag{10.137}$$

$$= \underbrace{\begin{bmatrix} 1 & 0 & 0 \end{bmatrix}}_{H} \hat{x}_k - w_k \tag{10.138}$$

can then be used as a "pseudomeasurement" in a Kalman filter to estimate x. The true altitude estimate will be

$$\hat{h}_k^{[\mathrm{TRUE}]} = \hat{h}_k^{[\mathrm{INS}]} - \hat{x}_{k,\,1}, \tag{10.139}$$

where $\hat{x}_{k,\,1}$ is the first component of the state vector $\hat{x}_k$ of Equation 10.120.

Characterizing Altitude Error Versus Accelerometer Error 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 10.28 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. This shows the trade-off between altitude error and the quality (and cost) of the vertical channel accelerometer.

The MATLAB m-file `AltStab.m` on the companion Wiley web site 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 10.28. These are Monte Carlo simulations, so the results should be different each time it is run.

10.5.1.4 Tightly Coupled Implementations These use a full INS error propagation model, including the effects of sensor compensation errors. As a minimum, they estimate and correct any changes in the navigation solution, as well as any changes in the sensor compensation parameters.

The stochastic model for unpredictable vehicle dynamics is replaced by the INS, which measures vehicle dynamics. However, the full navigation solution now includes INS attitude and sensor compensation parameters, which add state variables to be estimated. GNSS/INS integration generally increases the computational load.

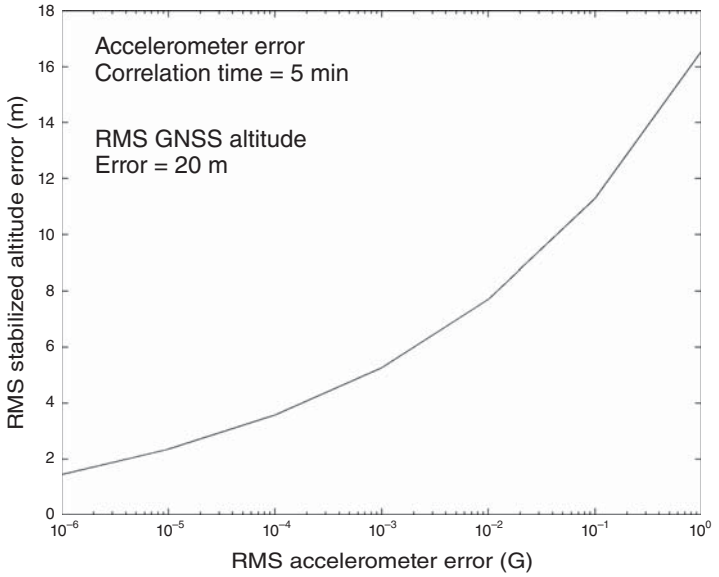


Figure 10.28 GNSS-aided INS altitude uncertainty versus RMS accelerometer error.

Keep in mind that there is no universal one-size-fits-all model for integrated GNSS/INS navigation, because there is no universal GNSS or INS. Although the underlying navigation principles may be universal, each physical realization has its own error characteristics, and each has its own models for those error characteristics. There are fundamental differences between single-frequency and dual-frequency GNSS receivers, for example, and there are fundamental differences between inertial navigators with different sensor suites. For example, Figure 10.29 shows some possible structures of dynamic coefficient matrices in Kalman filters for GNSS-only navigation (left column) and integrated GNSS/INS navigation (right column), and for single-frequency GNSS receivers (top row) and dual-frequency GNSS receivers (bottom row). Furthermore, the particulars of the F-matrix for integrated GNSS/INS navigation will depend very much on the sensor qualities and layout (e.g., gimbaled or strapdown) of the INS.

10.5.1.5 Benefits of 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.

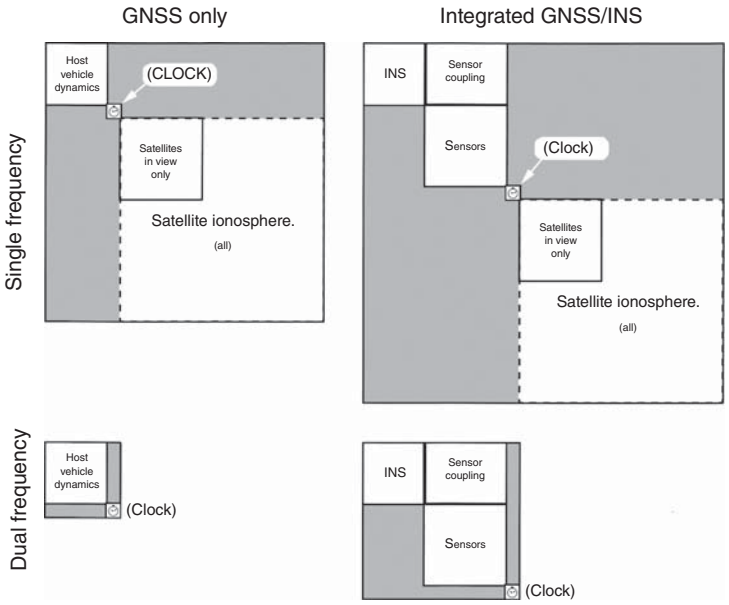


Figure 10.29 Dynamic coefficient matrices for GNSS and integrated GNSS/INS.

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 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.

10.5.2 MATLAB Implementations

In order to illustrate the general approach to integrated GNSS/INS navigation, we present here some specific examples, based on the models derived in Sections 10.3 and 10.4, using the dynamic simulators described in Section 10.2.7.

10.5.2.1 Gimbaled INS/GNSS Integration

Example 10.5 (Gimbaled INS and Dual-Frequency GPS Receiver) This example uses the same INS and GNSS models used for INS-only and GPS-only navigations with the same 100-km figure-8 test track simulator, so that the three navigation modes can be compared.

The MATLAB m-file `GNSSINS1.m` uses a matrix structure like that shown in the lower right in Figure 10.29 for the F matrix of its dynamic model, and the same process noise values used for medium accuracy INS and dual-frequency GPS in previous examples. The resulting plot of RMS horizontal position uncertainty for a 4-h excursion on the 100-km figure-8 test track simulator is shown in Figure 10.30. The uncertainty level shifts during periods when GNSS signals are available are likely due to DOP changes as satellites come into and go from view.

For comparison, Figure 10.31 is a multiplot of RMS horizontal position uncertainties from all three navigation modes:

1. Dual-frequency GPS only.
2. Medium accuracy INS.
3. Both the above, integrated.

In all three cases, there was one minute without GPS signals near the end of each hour. The integrated GPS/INS shows a reduced effect from those 1-min outages.

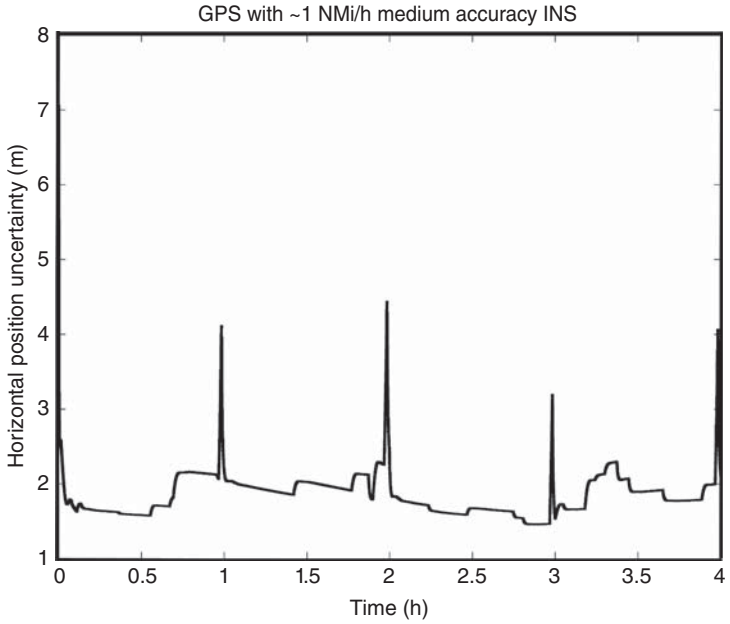


Figure 10.30 Integrating 1 NMi/h INS with dual-frequency GPS.

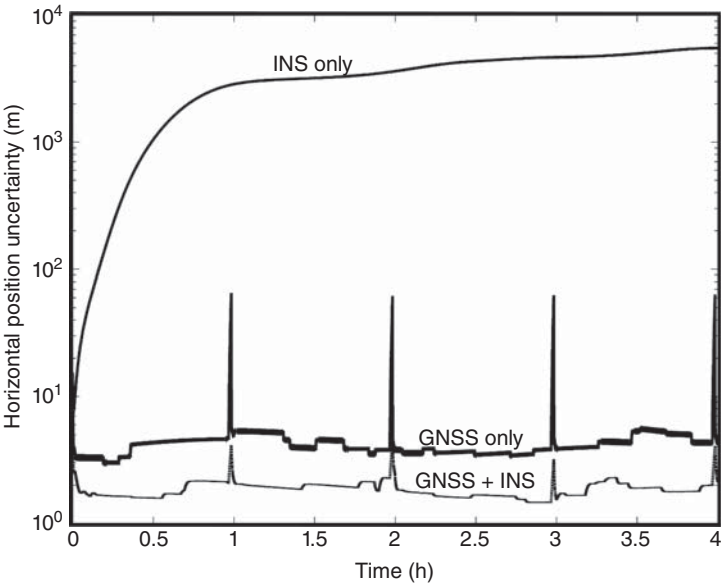


Figure 10.31 Comparing three navigation modes.

Note that the outages result in horizontal errors in the order of 5-m RMS horizontal error build-up for GNSS/INS integration, but in the order of 60-m RMS horizontal error build-up for GNSS-only navigation—an improvement of more than an order of magnitude. In the GNSS-only case, it is the host vehicle dynamics model that determines how fast navigation errors deteriorate when signals are lost.

10.5.2.2 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.[18]

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.

[18]Some few GNSS satellites may be in geostationary orbits. In order to maintain global coverage, however, most of the satellites will be in lower orbits with relatively high inclination angles to the equator.

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. In that case, attitude and attitude rate information from the INS can also aid the antenna switching implementation.

10.5.2.3 Model Modifications for Strapdown INS/GNSS Integration We have used a gimbaled INS with sensor axes aligned with ENU navigation coordinates in the derivations of INS error models for GNSS/INS integration.

As we have mentioned in Sections 10.4.4.3, 10.4.5.1, and 10.4.5.4, the error model modifications required for other INS implementations are related to the INS ENU models by the coordinate transformation matrix

$$C_{\text{NAV}}^{\text{SEN}}$$

from sensor-fixed coordinates to navigation coordinates—which is computed as an essential part of any INS implementation.

The dynamic coefficient matrix F_{CORE} for navigation error propagation remains the same as before, as does the dynamic coefficient matrix F_{SC} for errors in the sensor compensation parameters.

The only changes occur in the submatrices F_{24} and F_{35} of the dynamic coupling matrix between sensor compensation parameter errors and navigation errors, as illustrated in Figure 10.32.

These changes will depend on the particulars of the sensor compensation parameters. For the bias and scale factor compensation parameters used in the derivations

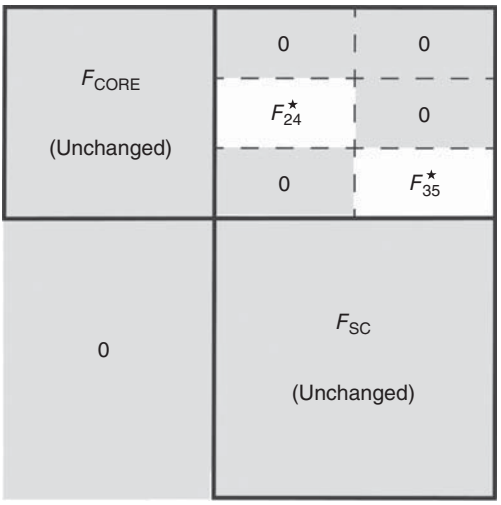


Figure 10.32 Changes to dynamic coefficient matrix for strapdown model. * Changed.

above, the changes are relatively simple. In that case, the new values are

$$F_{24}^{\star} = \begin{bmatrix} C_{\text{NAV}}^{\text{SEN}} & C_{\text{NAV}}^{\text{SEN}} D_{a^{\star}} \end{bmatrix} \tag{10.140}$$

$$a^{\star} = a_{\text{SEN}} \tag{10.141}$$

$$F_{35}^{\star} = \begin{bmatrix} C_{\text{NAV}}^{\text{SEN}} & C_{\text{NAV}}^{\text{SEN}} D_{\omega^{\star}} \end{bmatrix} \tag{10.142}$$

$$\omega^{\star} = \omega_{\text{SEN}}, \tag{10.143}$$

where a_{SEN} and ω_{SEN} are the compensated outputs of the respective (acceleration and rotation rate) sensors, and D_v represents a diagonal matrix with diagonal entries specified by the vector v.

For dynamic simulation in which the acceleration (a) and rotation rate (ω) are simulated in navigation coordinates, the simulated values

$$a_{\text{SEN}} = (C_{\text{NAV}}^{\text{SEN}})^{\text{T}} a_{\text{NAV}} \tag{10.144}$$

$$\omega_{\text{SEN}} = (C_{\text{NAV}}^{\text{SEN}})^{\text{T}} \omega_{\text{NAV}}, \tag{10.145}$$

where a_{NAV} and ω_{NAV} are the simulated dynamic conditions and the matrix $C_{\text{NAV}}^{\text{SEN}}$ (or its transpose) is generated as part of the attitude simulation.

As a rule, performance of GNSS/INS integrated navigation will be different for strapdown and gimbaled inertial systems with the same sensor performance specifications. The major differences are due to the different inputs the sensors will experience in the two (strapdown or gimbaled) implementations. The potential dynamic range of rotational inputs is generally much greater for strapdown systems, and strapdown systems also have more opportunities to sense part of the supporting forces used to counter gravity (the major sensible acceleration in many applications). As a consequence, strapdown systems may have greater observability of sensor compensation errors when a GNSS navigation solution is available.

Also, due to particulars of compensation parameter error dynamic coupling, carouseled or indexed gimbaled systems generally have better performance than systems maintaining a north alignment.

10.6 SUMMARY

1. Kalman filters have had an enormous impact on the accuracy, efficiency, and efficacy of navigation. Many of the newer navigation systems (e.g., GNSS or integrated GNSS/INS) were designed to use Kalman filters and were designed using Kalman filters.
2. When a satellite navigation system is being designed for a specific type of host vehicle (e.g., cargo ship, automobile, and fighter aircraft), its Kalman filter design can improve navigation performance by using a stochastic dynamic model tailored to the maneuvering capabilities of the intended host vehicle.

3. The design possibilities for integrating satellite navigation with inertial navigation are only 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 perform better than loosely coupled implementations.
4. We have provided detailed, illustrative examples of linear stochastic system models for GNSS and INS error dynamics and shown how these models behave under representative dynamic conditions.
5. We have derived some example models for specific types of inertial sensor errors. These do not begin to cover the range of possible error types encountered in inertial sensors, but they should serve as examples of how model derivations are done.
6. An error budget for any sensor system (including inertial navigators) is an allocation of the individual performances of its various subsystems so as to meet the overall performance of the system. This process allows some trade-offs, in which making some components less accurate can be compensated by making others more accurate. There may be overall performance standards for a navigation system, but this does not specify how these allocations of performances of subsystems is to be decided. This allows some freedom for minimizing costs or improving other performance metrics.
7. Real error budgets are based on real hardware with known, empirically derived error models. The example error budgets used here are strictly hypothetical, however. Sensor vendors and developers are better sources for sensor error characteristics, and well-vetted commercial MATLAB toolboxes for GNSS and INS navigation are likely sources for more reliable software.
8. 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.
9. The performance of integrated GNSS/INS navigation systems is not very sensitive to the stand-alone performance of the INS, which means that relatively 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 applications such as pilotless aircraft, driverless automobiles and trucks, automated surface mining, automated

control of farm equipment, and automated grading for roads. The potential economic payoff for this capability is substantial, and Kalman filtering will play an important part in this development.

10. GNSS and INS navigation problems provide good examples of Kalman filter applications to real-world physical systems, for which the Kalman filter models must be derived from the physics of the systems they represent, demonstrating Kalman's maxim that "Once you get the physics right, the rest is mathematics."

PROBLEMS

10.1 Equation 10.9 has formulas for vertical and horizontal dilution of precision. How would you break down horizontal dilution of precision (HDOP) into components for east dilution of precision (EDOP) and north dilution of precision (NDOP)?

10.2 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.

10.3 Run the MATLAB program `ClockStab.m` with clock stability values of 10^{-9}, 10^{-8}, 10^{-7}, and 10^{-6}. This is for the best of conditions: a stationary receiver with good satellite geometry. How is the estimate of clock drift C_d affected by clock stability under the best of conditions?

10.4 Run the MATLAB program `SatelliteGeometry.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? Can you explain why?

10.5 Answer the first three questions at the end of Example 10.3.

10.6 Why would gyrocompass alignment fail at the north and south poles?

10.7 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?

10.8 Add clock errors to the model in `GNSSshootoutNCE.m`. Assume 10^{-9} relative clock stability over one second. (See how it is done in Example 10.1.) Compare your results with those in Table 10.3.

10.9 Solve Equation 10.27 for P as a function of the elements of F and Q.

10.10 In Figure 10.11, RMS horizontal position error is bounded whenever GNSS signals are lost. Can you explain why?

REFERENCES

[1] L. J. Levy, "The Kalman filter: navigation's integration workhorse," *GPS World*, Vol. 8, No. 9, pp. 65–71, 1997.

[2] M. S. Grewal, A. P. Andrews, and C. G. Bartone, *Global Positioning Systems, Inertial Navigation and Integration*, 3rd ed., John Wiley & Sons, Inc., New York, 2013.

[3] A. Lawrence, *Modern Inertial Technology: Navigation, Guidance, and Control*, 2nd ed., Springer, New York, 1998.

[4] D. H. Titterton and J. L. Weston, *Strapdown Inertial Navigation Technology*, 2nd ed. IEE and Peter Peregrinus, UK, 2004.

[5] W. S. Widnall and P. A. Grundy, *Inertial Navigation System Error Models*, Technical Report TR-03-73, Intermetrics, Cambridge, MA, 1973.

[6] J. B.Y. Tsui, *Fundamentals of Global Positioning System Receivers: A Software Approach*, 2nd ed., John Wiley & Sons, Inc., New York, 2004.

[7] D. Sobel, *Longitude: The True Story of a Lone Genius Who Solved the Greatest Scientific Problem of His Time*, Penguin, New York, 1995.

[8] D. J. Allain and C. N. Mitchell, "Ionospheric delay corrections for single-frequency GPS receivers over Europe using tomographic mapping," *GPS Solutions*, Vol. 13, No. 2, pp. 141–151, 2009.

[9] J. Klobuchar, "Ionospheric time-delay algorithms for single-frequency GPS users," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. AES-23, No. 3, pp. 325-331, 1987.

[10] H. Dekkiche, S. Kahlouche, and H. Abbas, "Differential ionosphere modelling for single-reference long-baseline GPS kinematic positioning," *Earth Planets Space*, Vol. 62, 915–922, 2010.

[11] K. C. Redmond and T. M. Smith, *From Whirlwind to MITRE: The R&D Story of the SAGE Air Defense Computer*, MIT Press, Cambridge, MA, 2000.

[12] K. R. Britting, *Inertial Navigation Systems Analysis*, Artech House, Norwood, MA, 2010.

[13] P. G. Savage, *Strapdown Analytics*, Vol. 2, Strapdown Associates, Maple Plain, MN, 2000.

[14] M. Schuler, "Die Störung von Pendel-und Kreiselapparaten durch die Beschleunigung der Fahrzeuges," *Physicalische Zeitschrift*, Vol. B, p. 24, 1923.

[15] GPSoft, *GPSoft Inertial Navigation System Toolbox for Matlab*, Version 3.0, GPSoft, Athens, OH, 2007.

[16] S. F. Schmidt, "The Kalman filter—its recognition and development for aerospace applications," *AIAA Journal of Guidance, Control, and Dynamics*, Vol. 4, No. 1, pp. 4–7, 1981.

APPENDIX A

SOFTWARE

Software is like entropy. It is difficult to grasp, weighs nothing, and obeys the second law of thermodynamics; i.e. it always increases.
—Norman R. Augustine, Law Number XVII, p. 114, *Augustine's Laws,* AIAA, 1997.

A.1 APPENDIX FOCUS

This Appendix organizes and summarizes descriptions of the MATLAB® software used throughout the chapters and available online at the Wiley web site, `www.wiley.com/go/kalmanfiltering`, along with the necessary supporting software. These are 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 website. In most cases, comments within the scripts further describe the inputs, processing, and outputs.

A.1.1 Notice

Software fidelity is especially critical for navigation systems, because human lives and safety could be at risk.

The software provided with this book 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 merchantability for commercial purposes. These routines should not be used as-is for any purpose or

Kalman Filtering: Theory and Practice Using MATLAB®, Fourth Edition.
Mohinder S. Grewal and Angus P. Andrews.
© 2015 John Wiley & Sons, Inc. Published 2015 by John Wiley & Sons, Inc.

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.1.2 General System Requirements

The MATLAB scripts on the web site are designed for MATLAB environments. Information on MATLAB can be obtained from

The Math Works, Inc.	Tel:	508-647-7000
3 Apple Hill Drive	Fax:	508-647-7101
Natick, MA 01760 USA	e-mail:	info@mathworks.com
	Website:	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.1.3 Organization

The m-file descriptions are organized here by the chapters in which the supporting concepts are presented and organized within each chapter in alphabetical order. The Microsoft Word 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.

Because some of these MATLAB scripts call other scripts, users are advised to copy all MATLAB scripts to a common subdirectory of the MATLAB "work" directory on their own computers.

A.2 CHAPTER 1 SOFTWARE

`LeastSquaresFit.m` is an example of using least squares to fit a straight line through trending data. It is the method used in Chapter 10 for characterizing the error growth rates of inertial navigation systems (INSs).

`RMSHorINS1m.mat` is a data file borrowed from Chapter 10, containing values of root-mean-square (RMS) position errors for an INS. It is used by `LeastSquaresFit.m`.

A.3 CHAPTER 2 SOFTWARE

`expm1.m` demonstrates the MATLAB-callable matrix exponential function `expm` by taking exponentials of six randomly generated matrices, plus a shift matrix.

A.4 CHAPTER 3 SOFTWARE

`CumProbGauss.m` computes the cumulative probability function for the zero-mean unit-normal Gaussian distribution.

`GaussEqPart.m` partitions the domain of the zero-mean univariate Gaussian probability function into n segments, each of which has probability measure $1/n$, and returns a vector of the medians of those segments.

`meansqesterr.m` generates Figure 3.1, which is a multiplot of the mean-squared estimation error as a function of the estimated value—for five different probability distributions with a common mean (2) and variance (4):

1. Gaussian (or normal).
2. Lognormal.
3. Laplace.
4. Chi-squared with two degrees of freedom.
5. Uniform.

`pChiSquared.m` returns the probability density at x of the chi-squared distribution with given mean. (For chi-squared distributions, variance is always twice the mean.)

`pLaplace.m` returns the probability density at x of the Laplace distribution with given mean and variance.

`pLogNormal.m` returns the probability density at x of the lognormal distribution with given mean and variance.

`PlotpYArctanaX.m` generates Figure 3.3 and a mesh plot showing how the arctangent-normal distribution shape varies with scaling—going from unimodal to bimodal. Calls `pYArctanaX.m`.

`pNormal.m` returns the probability density at x of the normal distribution with given mean and variance.

`pUniform.m` returns the probability density at x of a uniform distribution with given mean and variance.

`pYArctanaX.m`, given a and X, computes the probability density function of $Y = \arctan(aX)$, where X has a zero-mean unit-normal univariate distribution.

A.5 CHAPTER 4 SOFTWARE

`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).

The MATLAB Signal Processing Toolbox also has many utilities for the generation and analysis of random sequences.

A.6 CHAPTER 5 SOFTWARE

`exam57.m` is a MATLAB script for demonstrating Example 5.7 in MATLAB. Demonstrates a Kalman filter estimating the state (position and velocity) of a damped harmonic oscillator with constant forcing.

`exam58.m` is a MATLAB script for demonstrating Example 5.8 in MATLAB, for radar tracking of 6DOF airborne vehicle using range, bearing, and elevation measurements. Plots histories of six mean-squared state uncertainties and six magnitudes of Kalman gains for intersample intervals of 5, 10, and 15 s.

`F2Phi.m` demonstrates KF execution errors due to approximations in converting a dynamic coefficient matrix F to a state-transition matrix Φ.

`obsup.m` is a MATLAB script implementation of the Kalman filter observational update, including the state update and the covariance update (Riccati equation).

`ProbCond.m` demonstrates conditioning of a Gaussian probability distribution of a random walk variate due to taking noisy measurements. Plots history of probability density functions over time, to demonstrate how measurements tighten the distribution, which otherwise diffuses and broadens.

`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 CHAPTER 6 SOFTWARE

`BMFLS.m` implements the complete Biswas–Mahalanabis fixed-lag smoother, through the transient phase from the first measurement and through the transition to steady-state state augmentation, and beyond.

`FiniteFIS.m` computes and plots the "end effects" of a fixed-interval smoother.

`FIS3pass.m` fixed-interval 3-pass smoother demonstration. Simulates a trajectory for a scalar linear time-invariant model driven by pseudorandom measurement noise and dynamic disturbance noise, applies a 3-pass-fixed interval smoother to the resulting measurement sequence, and plots the resulting estimates along with the true (simulated) values and computed standard deviations σ of estimation uncertainty.

`FPSperformance.m` computes the expected performance of fixed-point smoothing as a function of time, up to and beyond the fixed time at which the estimate is to be estimated, and plots results.

`RTSvsKF.m` is a MATLAB script for Monte Carlo simulation the state variable estimates of a random walk process, 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.

A.8 CHAPTER 7 SOFTWARE

`shootout.m` provides a demonstration of the relative numerical stability of nine different ways to perform the covariance correction on Example 7.2. To test how different solution methods perform as conditioning worsens relative to the machine-dependent roundoff error ε, 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 $\delta \to \varepsilon$. 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.

`carlson.m` performs the Carlson "fast triangular" implementation of the Kalman filter measurement update.

`housetri.m` performs Householder upper triangularization of a matrix.

`joseph.m` performs the Joseph-stabilized implementation of the Kalman filter measurement update, as proposed by Joseph [1].

`josephb.m` performs the Joseph-stabilized implementation of the Kalman filter measurement update, as modified by Bierman.

`josephdv.m` performs the Joseph-stabilized implementation of the Kalman filter measurement update, as modified by DeVries.

`potter.m` performs the Potter "square-root" implementation of the Kalman filter measurement update.

`Schmidt.m` performs the Schmidt–Householder temporal update of the triangular Choleksy factor of the a posteriori covariance matrix of estimation uncertainty.

`SPDinIP.m` performs in-place inversion of symmetric positive definite matrix using UD decomposition.

`SPDinv.m` performs inversion of symmetric positive-definite matrix using *UD* decomposition.

`thornton.m` implements a temporal update of the UD filter due to Thornton.

`UD_decomp.m` performs UD decomposition of a symmetric positive-definite matrix in-place.

`UDinv.m` performs inversion of UD decomposition factors U and D of a symmetric positive-definite matrix.

`UDInvInPlace.m` tests the accuracy of in-place algorithms `UD_decomp.m`, `UDinv.m`, and `SPDinvIP.m` on randomly generated symmetric positive-definite matrices of dimension 1×1 to 12×12.

`utchol.m` performs upper triangular Cholesky factorization for initializing the Carlson fast triangular implementation of the Kalman filter measurement update.

`udu.m` performs UDU ("modified" Cholesky) decomposition of a symmetric positive-definite matrix P as $P = UDU^T$, where U is unit upper triangular and D is diagonal.

`VerifySchmidt.m` verifies the Schmidt–Householder temporal covariance update function `Schmidt.m` using pseudorandom matrices $A(n,n) = \Phi C_P$, $B(n,r) = GC_Q$ for $1 \le n \le 12$, $1 \le r \le 12$, by computing C = Schmidt(A,B) and the relative error as `max`(max(abs(A*A'+B*B'-C*C')))/max(max(abs(C*C'))).

A.9 CHAPTER 8 SOFTWARE

`ArctanExample.m` works Example 8.11 and plots Figure 8.21, comparing extended Kalman filter and unscented Kalman filter approximations of arctangent of a scaled unit-normal distribution as the scaling varies over several orders of magnitude.

`dhdxarctan.m` extended Kalman filter approximation of arctangent sensitivity matrix as its partial derivative with respect to the state variable.

`EKFDampParamEst.m` implements and demonstrates an extended Kalman filter estimating the damping parameter of a damped harmonic oscillator driven by white noise.

`ExamIEKF.m` performs Monte Carlo analysis of iterated extended Kalman filtering on a nonlinear sensor problem.

`harctan.m` arctangent measurement function.

`hCompSens.m` measurement as a function of the sensor input, the sensor scale factor, and the sensor output bias.

`NonLin.m` n-dimensional sensor output vector as quadratic function of sensor input vector.

`utchol.m` upper triangular Choleksy decomposition.

`UTsimplex.m` implements the lowest complexity unscented transform using "simplex" sampling.

`UTsimplexDemo.m` is a MATLAB demonstration of its application of `UTsimplex.m`.

`UTscaled.m` implements the scaled unscented transform. `UTscaledDemo.m` demonstrates its use.

`UTscaledDemo.m` demonstrates the use of `UTscaled.m`.

A.10 CHAPTER 9 SOFTWARE

`InnovAnalysis.m` Innovations analysis [1] with model parameters deliberately scaled up and down by a factor of two–to demonstrate its effect on weighted squared innovations.

`KFvsSKF.m` compares performance of Schmidt–Kalman filter versus Kalman filter on problem of estimating the state of a damped harmonic resonator excited by white noise, using measurements of resonator displacement corrupted by white noise plus exponentially correlated bias.

`SKF_Obs.m` implements Schmidt–Kalman observational update.

`SKF_Temp.m` implements Schmidt–Kalman temporal update.

A.11 CHAPTER 10 SOFTWARE

`AltStab.m` simulates and plots time histories of RMS stabilized and "unstable" [i.e., unaided by global navigation satellite system (GNSS)] altitude error for a range of RMS accelerometer noise levels.

`ClockStab.m` Given relative RMS GNSS receiver clock stability over a 1-s intervals, simulates 10 min of filtering with fixed satellite geometry and plots histories of RMS position and clock uncertainties.

`DAMP3ParamsD.m` performs the same function as `MODL6Params.m`.

`Fcore4.m` performs the same function as `Fcore9.m`, except latitude is entered in radians (not degrees).

`Fcore7.m` performs the same function as `Fcore9.m`, but without the error state variables for altitude and altitude rate. Output is the 7×7 dynamic coefficient matrix for the resulting 7-state navigation error model.

[1] Note: Signal Processing Toolbox has routines for computing autocorrelation of innovations to test white noise hypothesis, and some of these routines have counterparts for GNU Octave, a MATLAB-like environment in the public domain.

`Fcore9.m` Given the latitude of the host vehicle and its velocity and acceleration in ENU coordinates, computes the 9×9 dynamic coefficient matrix for the nine core navigation error variables.

`Fig8Mod1D.m` is called by `GNSSshootoutNCE.m` for the host vehicle dynamic model with one degree of freedom (distance along track).

`Fig8TrackSimRPY.m` Figure-8 track dynamic simulator in Roll-Pitch-Yaw coordinates. Given the time (t), returns the vehicle position in east-north-up (ENU) coordinates, plus the vehicle roll, pitch, and yaw angles, the sensed acceleration and rotation rates in roll-pitch-yaw (RPY) coordinates, and the transformation matrix $C_{\text{RPY}}^{\text{ENU}}$ from ENU to RPY coordinates.

`GNSSINS1.m` performs simulation and performance analysis of GPS integrated with an INS with about 1 nautical mile/h circular error probable (CEP) rate.

`GNSSINS123.m` performs covariance analyses simultaneously for GPS integrated with three inertial navigators with 0.1, 1, and 10 nautical miles/h CEP rates. Test conditions on a 100-km figure-8 test track with GPS signal outages 1 min out of every hour for 4 h.

`GNSSINS1230.m` performs the same functions as `GNSSINS123.m`, but with total GNSS signal outage for the last hour of testing.

`GNSSshootoutNCE.m` generates the GNSS navigational performance statistics shown in Table 10.3 for the host vehicle models shown in Table 10.2. It assumes no clock errors.

`GPS1FreqOnly.m` simulates navigation with a single-frequency GPS receiver on a 100-km long figure-8 test track for 4 h, with signal outages for 1 min each hour. Includes the uncompensated Klobuchar compensation residuals as state variables. Plots RMS navigation statistics versus time.

`GPS2FreqOnly.m` simulates navigation with a dual-frequency GPS receiver on a 100-km long figure-8 test track for 4 h, with signal outages for 1 min each hour. Plots RMS navigation statistics versus time.

`HSatSim.m` is called by `GNSSshootoutNCE.m` to obtain simulated GPS satellite sightings. Needs the global parameter arrays *RA* and *PA* computed by `YUMAdata.m`.

`INSErrBud0point1NMiPerHr.m` Performs error budget analysis of an INS with about 0.1 nautical mile/h CEP rate. Outputs are the effective rescaling of CEP rate attainable by scaling budgeted RMS sensors errors up and down by a factor of 10.

`INSErrBud1NMiPerHr.m` performs the same functions as `INSErrBud0point1NMiPerHr.m`, but for an INS with about 1 nautical mile/h CEP rate.

`INSErrBud10NMiPerHr.m` performs the same functions as `INSErrBud0point1NMiPerHr.m`, but for an INS with about 10 nautical miles/h CEP rate.

`MediumAccuracyINS.m` performs covariance analysis of an INS (with about 1 nautical mile/h CEP rate) using a barometric altimeter to stabilize vertical navigation errors.

`MODL5Params.m` Given the standard deviations of acceleration (σ_{acc}) and velocity (σ_{vel}), plus the exponential correlation time of acceleration (τ_{acc}) along a

single axis and the discrete time-step Δt, returns the Kalman filter matrix parameters Φ, Q, and P_0 for the fifth host vehicle dynamic model in Table 10.2.

`MODL6Params.m` Given the standard deviations of acceleration (σ_{acc}), velocity (σ_{vel}), and position (σ_{pos}), plus the exponential correlation time of acceleration (τ_{acc}) along a single axis and the discrete time-step Δt, returns the Kalman filter matrix parameters Φ, Q, and P_0 for the sixth host vehicle dynamic model in Table 10.2.

`ReadYUMAdata.m` transforms an ascii file downloaded from `www.navcen.uscg.gov-/ftp/GPS/almanacs/yuma/` into an m-file `YUMAdata.m` used for GPS satellite geometry simulations. This allows users to obtain more current GPS satellite configurations.

`SatelliteGeometry.m` requests inputs for the azimuths and elevation angles to four satellites, then simulates 10 min of filtering with that geometry. Plots histories of RMS position and clock uncertainties.

`SchmidtKalmanTest.m` evaluates performance degradation in terms of the penalty in RMS position error incurred by using the Schmidt–Kalman filter during a simulated dynamic test with a single-frequency GPS receiver.

`YUMAdata.m` MATLAB-callable m-file for initializing arrays of right ascensions (`RA`) and in-plane locations of satellites at the reference time. Used for simulating GPS constellations, assuming circular orbits with 55° inclinations. The MATLAB function `HSatSim.m` uses these global arrays to generate pseudorange measurement sensitivity matrices for GPS receivers at a given latitude, longitude, and time.

A.12 OTHER SOFTWARE SOURCES

The MATLAB programming environment is well suited for developing and verifying Kalman filtering applications, and there are already many MATLAB "toolboxes" specialized for Kalman filtering. A few of them are listed here. In addition, The MathWorks web site is a good source for examples and tutorials on Kalman filtering and its applications.

A.12.1 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 Signal Processing Toolbox

Also available from The Mathworks, the Signal Processing Toolbox includes routines for frequency-domain analysis of sensors and can be used for characterizing and synthesizing dynamic models from dynamic measurements. It is especially useful for

the analysis of data from inertial measurement units to uncover exploitable dynamic characteristics of vehicles used in applications of interest.

A.12.3 GNU Octave and Free Software Foundation

GNU Octave is a collection of MATLAB-like replacement software available for free downloading. For more information, see their web site at `http://www.gnu.org/software/octave` As mentioned in Chapter 3 (for the `chi2pdf` script), contributors to the Free Software Foundation offer many MATLAB scripts allegedly performing the same functions as scripts with the same names in MATLAB toolboxes. If available for free downloads, these can usually be found a Google search on the MATLAB m-file or function name.

A.12.4 Software for Kalman Filter Implementations

A.12.4.1 Transactions on Mathematical Software (TOMS) 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. These are not MATLAB toolboxes or MATLAB scripts, but they are good sources for robust computer implementations of Kalman filtering.

A.12.5 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., References [3] or [2]) that come with the appropriate code on machine-readable media.

A.12.6 GNSS and Inertial Navigation Software

There are numerous sources of software designed specifically for the design and analysis of INS and GNSS . We cite here just a few examples.

A.12.6.1 Aided Inertial Navigation System (AINS) The Geomatics group at the University of Calgary, Canada, offers software and services for GNSS simulation and analysis, including the Aided Inertial Navigation System (AINS) Toolbox for MATLAB.

A.12.6.2 Aalto University Many sources for the unscented Kalman Filter can be found by searching the World Wide Web. There is an *EKF/UKF Toolbox for MATLAB* made available by the Aalto University School of Science, Department of Biomedical Engineering and Computational Science (BECS), Otaniemi Campus, Espoo, Finland at http://www.mathworks.com/matlabcentral/fileexchange/.

GPSoft. The MATLAB toolboxes

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

are produced by GPSoft (http://www.gpsoftnav.com) and marketed through

Navtech Seminars & GPS Supply
www.navtechgps.com
Tel: 1-703-256-8900
Fax: 1-703-256-8988

Many other commercial sources can be located by searching the World Wide Web.

REFERENCES

[1] Bucy, R.S. and Joseph, P.D. (2005) *Filtering for Stochastic Processes with Applications to Guidance*, American Mathematical Society, Chelsea Publishing.

[2] Kahaner, D., Moler, C., and Nash, S. (1989) *Numerical Methods and Software*, Prentice-Hall.

[3] Press, W.H., Flannery, B.P., Teukolsky, S.A., and Vettering, W.T. (2007) *Numerical Recipes in C++: The Art of Scientific Computing*, 3rd edn, Cambridge University Press.

INDEX

a posteriori, 29
a priori, 29
Aalto University, 603
acceleration, 545, 548
  gravitational, 548, 551
  sensor, 548
accelerometer, 546, 548
  gyroscopic, 548
  integrating, 548
  proof mass, 548
Adrian, Robert, 8
affine Kalman filter, 370–371
affine transformation, 95–96, 371, 567
AFOSR, 16
Agee, William S., 22, 35
algebraic equation, 141, 144
  Riccati, 202, 210–219
    solution, 213–219
aliasing, 444
alignment (INS), 553
  gyrocompass, 553–554
  transfer, 556
  velocity matching, 556
Allen, Edward, 112
altitude instability, 565
AltStab.m, 582, 599
Anderson, Brian D. O., 171, 243, 244, 256
Anderson-Chirarattananon bound, 244
Andrews, Geraldine, xiii
ArctanExample.m, 598
argmax, 179
autocorrelation, 122
autocovariance, 122
autonomous systems, 47

Bachelier, Louis, 113
Balakrishnan, A. V., 18
Baras, John S., 26
Bass, Robert W., xiii, 16, 17, 25
Battin, Richard H., 6, 16, 18
Bayes rule, 185
Bayes, Thomas, 6, 14
Beidou, 512
Bellman, Richard E., 16
Beneš, Václav, 417
Beneš filter, 417–418
  Performance vs EKF, 418
Bennet, J. M., 21, 34

Kalman Filtering: Theory and Practice Using MATLAB®, Fourth Edition.
Mohinder S. Grewal and Angus P. Andrews.
© 2015 John Wiley & Sons, Inc. Published 2015 by John Wiley & Sons, Inc.

Bernoulli, Jakob, 6, 14
bias
  clock, 516, 518, 520, 527–538
  gyro, 555
  sensor, 557
Bierman, Gerald J., 6, 22–24, 34, 293, 323, 326, 351
Bierman rule,323
bierman.m, 326, 597
Birkhoff, George D., 119
Biswas, K. K., 256, 258
Biswas-Mahalanabis smoother, 258–264
BMFLS, xv, 258–264
BMFLS.m, 596
Bode, Hendrik W., 131
Bode, Johann, 7
Bohr, Niels, 169
Boykow, Johann M., 545
bps, xv
Brownian motion, 113
BSAC, 548
Bucy, Richard S., 6, 16–18, 25, 27, 171

CalTech, 22
Campbell, L. A., 25
Cao, Yi, 327
Cardan suspension, 550
Cardano, Girolamo, 6, 14, 555
Carlson, Neal A., 6, 22
carlson.m, 597
carouseling, 550–551, 571
Cauchy sequence, 434
Cauchy, Augustin L., 434
CDMA, xv, 515
central moment, 82–85
CEP, xv, 507
  rate, 565
Ceres, 4, 7–8
chance, 111
characteristic function, 80
characteristic vector, 218–223
chi2pdf.m, 104
Chirarattananon, Surapong, 244
chi-squared test, 104, 396
Cholesky, 6
  decomposition, 21
    augmented, 92
  factor, 21, 294–295
    generalized, 21–22, 295
    modified, 295
    triangular, 295
  least-squares, 92
Cholesky, André-Louis, 21
CIGTF, 508
clock
  bias, 516, 520
  drift, 506
  error models, 527–530
  flicker noise, 528
  measurement sensitivity, 530
  synchronization, 513–517, 527
  syntonization, 530–531
ClockStab.m, 533, 599
closed-form solution, 56
Cohen, E. Richard, xiii
companion form, 46, 49
Compass, 392
computational complexity, 484–487
computer
  flightworthy, 545
  navigation, 549
conditioning, 27
conformable, 94
controllability, 65–66
convolution integral, 125
coordinates
  ENU, 517, 554, 558
  inertial, 550
  locally level, 547, 557
  navigation, 547, 550–556, 559
  NED, 535, 543
  rotating, 553
  RPY, 558–559
  sensor, 559
Coriolis effect, 551, 553
  gyroscope, 547–558
correlated noise, 200
correlation, 86
  coefficient, 86, 332–334
covariance
  analysis, 435, 507
  matrix, 82
    repair, 455
cross-covariance, 84
cruise navigation, 565
CumProbGauss.m, 595
cumulative probability, 79
cybernetics, 15

D'Alembert, Jean le Rond, 171
Damp3ParamsE.m, 527, 599
Dang, Dean, xiii
data rejection, 457–460
datum, 534
de Moivre, Abraham, 6, 14
De Vries, Thomas W., xiii
Decomposition
  Cholesky, 21
  QR, 23
  singular value, 180
delta function, 118
density function, 74, 77
detection, 27
  Schweppe, 105, 396
  vs tracking, 418
dhdxarctan.m, 598
differential equation, 40, 42
  closed-form solution, 56
  companion form, 46
  homogeneous, 45–49, 51–52
  Laplace transform 56–58
  linear, 37, 39, 47–50
  nonhomogeneous, 45–46, 49, 55
  Riccati, 199
  stochastic, 113, 125, 136
  time-invariant, 56
dilution of precision, 516–517
Dirac delta, 78, 81, 118, 126
displacement rank, 262
distribution, 76
  Cauchy, 81
  Chi squared, 90, 104
  Gaussian, 77–78, 90
    cumulative, 79
  moments, 81–85
  multivariate, 78
  Laplace, 90
  lognormal, 90
  normal, 78, 90
  uniform, 90
domain partitioning, 396–398
DOP, 516–517
double factorial, 84
doubling method, 488–491
Draper, Charles Stark, 546, 555, 565
Draper Laboratory, 18, 548
dual-frequency GNSS, 515, 530, 539
duality, 65
dual-state filter, 467–471, 494
Durrant-Whyte, 405
Dyer, Peter, 22
dynamic coefficient matrix, 48
  companion form, 46, 49–50
dynamic system, 37, 39–40, 42
  continuous time, 39, 46–50
  discrete time, 39, 46–47, 51, 59–61
  linear, 37, 39, 47–50
  models, 39
  observable, 61–64
  rotational, 43
  time-invariant, 47
  time-varying, 47, 56
  unobservable, 61

East-North-Up coordinates, 517, 554, 558
Einstein, Albert, 15, 26
EKF, xv, 18, 25, 374
EKFDampParamEst.m, 598
EKF/UKF Toolbox, 603
ENU, xv, 426
erf, 79
ergodic process, 119
error budget, 491–495, 557–558
error function, 79
estimate, 87
  least mean squared, 87–91
  least squares, 6, 8, 87
  optimal, 88
  recursive, 91–92
  squared error, 88
estimation problem, 169
evolution operator, 53
examIEKF.m, 598
exam57.m, 596
exam58.m, 596
expectancy, 73, 79
  operator, 74, 80
expected value, 80
expm, 57
expm1.m, 594
exponential of a matrix, 56–57
exponentially correlated process, 122–123
extended Kalman filter, 18, 25, 374, 471

F2Phi.m, 596
factorization methods, 23–24
Fcore4.m, 599

Fcore7.m, 563, 599
Fcore9, 563, 600
FDMA, xv, 515
Fermat, Pierre de, 6, 14
Fermi, Enrico, 26
Fertig, Kenneth W., xiii
fiber-optic gyroscope, 547
Fig8Mod1D.m, 600
Fig8TrackSimRPY.m, 511, 600
figure-8 track, 509–512, 532–539
  dynamics, 510–511
filter, 169
  anti-aliasing, 153
  etymology, 3
  Kalman, 1, 169, 172, 503–505, 518–532
    affine, 370–371
    diagnostics, 428–456
    divergence, 434–435
    engineering, 427
    extended, 18, 25, 374, 471
    history, 16–17
    linear regression, 405
    monitoring, 433–434
    roundoff, 284
    unscented, 26, 30, 405–416
  Kalman–Bucy, 197
  nonlinear, 24–27, 367
  particle, 26, 401–402
  radar tracking, 225–228
  Schmidt-Kalman, 374, 471, 540–542
  sequential Monte Carlo, 26
  shaping, 201, 227
  sigma-point, 26, 402
  sigmaRho, 330–346, 416
  square root, 21–23
  tracking, 522
  unscented, 318
  Wiener, 201–202
  Wiener-Kolmogorov, 8, 15, 17, 20
FiniteFIS.m, 596
Fisher, Ronald A., 6, 10, 177
Fisher information matrix, 10
FIS3Pass.m, 596
fixed-interval smoother, 241, 244–256
fixed-lag smoother, 241–242, 256–268
fixed-point smoother, 241–242, 268–274
flat priors, 179
flicker noise, 528
FOG, 547
Fokker, Adriaan, 26, 417
Fokker–Planck equation, 26
  conditioned, 27, 418
F2Phi.m, 596
FPSperformance.m, 596
freeway model, 379–386
functional, 79
  linear, 79–81, 103–104
  sampling, 81
fundamental solution, 52–57
  existence, 53
  nonsingular, 53
  properties, 53
fuzzy systems theory, 16

Gaffney, Rev. Joseph, xiii
gain scheduling, 487–488
Galilei, Galileo, 6
Gamow, George, 565
Gauss, Carl F., 6–8, 11–12, 113, 239
GaussEqPart.m, 595
Gaussian distribution, 77–78, 176
Gaussian maximum likelihood, 176
Gaussian process, 118, 120–121
GDOP, 517
Gelb, Arthur, 29, 201
Gentleman, William M., 24
Gibbs, Josiah W., 119
Gieseking, Darrell L., 6
Givens, James W., 24
GLONASS, 512, 514–515
GMLE, xv, 176
GNSS, xv, 240, 242, 504, 506
  Compass, 512
  dilution of precision, 516–517
  dual-frequency, 515, 530, 539
  error modeling, 518–532
  Galileo, 512
  GLONASS, 512, 515
  GPS, 512–514
  ground segment, 512
  history, 510–512
  host vehicle, 506
  multipath, 510, 517–518
  navigation, 392
    Doppler, 512–513
    error sources, 513
    timing-based, 513
  navigation solution, 534–540

propagation delays, 514–515
dual-frequency, 515
Klobuchar model, 514
pseudorange, 180–181, 392, 506, 518–520
differences, 542
receiver antenna, 506, 512, 514, 549
receiver clock, 515–516
errors, 515
receiver noise, 518
simulation, 533–534
space segment, 512–514
GNSSINS1.m, 585, 600
GNSSINS123.m, 600
GNSSINS1230.m, 600
GNSS/INS integration, 506, 508, 578–588
GNSSshootoutNCE.m, 510, 537, 539, 600
GNU, 104, 599, 602
Goddard, Robert H., 16
Gold, Robert, 515
Gold codes, 515
GPS, xv, 512–515, 531–534
ephemerides, 514, 533–534
Selective Availability, 515
GPS1FreqOnly.m, 600
GPS2FreqOnly.m, 539–540, 600
GPSoft INS Toolbox, 564, 603
Gram, Jørgen P., 10
gramian, 9–11
gravitational acceleration, 548–549, 551, 556
gravitational field, 548
gravity, 547, 548
Grewal, Sonja, xiii
guidance, 549
Guier, William, 10
Gunckel, Thomas L. II, xiii
gyro, 547
gyrocompass alignment, 553–554
gyroscope, 547–548
bias, 555, 569
Coriolis, 547–558
displacement, 547
fiber-optic, 547
laser, 547
momentum wheel, 547
rate, 547
scale factor, 555, 569
whole-angle, 547
Hanson, Richard J., 284
harctan.m, 598
harmonic oscillator, 44–46, 54, 123, 143, 146, 149
disturbance noise, 146
underdamped, 54, 123, 143, 146, 149
Harrison, John, 513
hCompSens.m, 599
HDOP, 517
health monitoring, 40, 429, 433
Heckman, Dwayne W., xiii
Hegel, Georg, 7
Hemes inversion formula, 220
Herschel, Friedrich, 7
homogeneous equation, 45–49, 51–52
host vehicle, 506
dynamics, 509, 520–527
simulator, 509–510, 537–539
Householder, Alston S., 22, 24
housetri.m, 597
HSatSim.m, 534, 600
Hubbs, Robert A., xiii
Huygens, Christiaan, 6, 14

IBM, 284
identification problem, 26–27, 418
IEKF, xvi, 385–389
i.i.d. process, 114–115
IIR, xv, 153
importance sampling, 402
IMU, 549
indexing (INS), 550
inertia, 547
inertial
measurement unit, 549
navigation, 505
history, 505
navigation system, 544–578
gimbaled, 550–553
hybrid, 551
strapdown, 550, 553
platform, 549
reference frame, 547
reference unit, 549
sensor, 547
acceleration, 548
accelerometer, 548
assembly (ISA), 548
gyroscope, 547

inertial (*Continued*)
input axis, 548
MEMS, 505, 546, 550
multi-axis, 548
rotation, 547
information matrix, 10, 63, 176–186
addition, 182
Fisher, 10
inverse, 179
singular, 176
singular values, 180
svd, 180
initialization, 396
partitioning, 396–398
innov02, 349
InnovAnalysis.m, 599
innovations, 343, 396, 428–432
covariance, 343, 346, 429
notation, 428
properties, 429
input axis, 548
INS, xvi, 242, 544
alignment, 553
gyrocompass, 553–555
on-the-fly, 556
transfer, 556
altitude stabilization, 565, 572
calibration, 557, 566–567
carouseled, 550–551, 571
compensation parameters, 557, 567–571
computer, 549
error budget, 557–558, 572
gimbaled, 550
hybrid, 551
initialization, 553
attitude, 553–555
performance, 564–565
underway, 556
strapdown, 550–553, 557
transfer alignment, 556
user interface, 549
vestibular system, 545
INSErrBud1NMiPerHr.m, 574, 575, 577, 600
INSErrBud10NMiPerHr.m, 574, 575, 600
INSErrBud0point1NMiPerHr.m, 574, 575, 577, 600
instrument cluster, 549
interpolation, 240
interpolative reasoning, 16
Inverarity, Gordon, xiii
inverse Laplace transform, 56–57
IRU, 549
ISA, xvi, 548–551
Itô, Kiyosi, 6, 27, 114, 125, 146
Itô calculus, 114

Jacobian matrix, 372–373
Jet Propulsion Laboratory, 22
Joseph, Peter D., 21, 175
joseph.m, 597
josephb.m, 597
josephdv.m, 597
Jover, Juan-Manuel, 23
JPL, 22
Julier, Simon J., xiii, 405

Kailath, Thomas, xiii, 6, 22, 191, 256, 262, 276, 428
Kain, James, xiii
Kalman, Rudolf E., xiii, 16, 65
Kalman filter, 1, 172–197, 231, 512
advantages, 20
affine, 370–371
application, 4–5, 18
approximation, 372–398
architecture, 504
Bierman–Thornton, 322–330, 485
computer requirements
memory, 478–484
throughput, 467, 484–485
wordlength, 447, 461, 478, 480
convergence, 434, 436–437
cost, 433–434, 461, 485
covariance analysis, 435
data rejection, 457, 460
diagnostics, 428–456
discovery, 16
divergence, 434–435
dual-state, 467–471, 494
dynamic model, 170, 523
engineering, 331
equations, 190
extended, 18, 25, 374, 471
health monitoring, 40, 429, 433
impact, 19
implementation, 18, 20
initialization, 396

innovations analysis, 428–432
introduction, 18
iterated, 385–389
linear regressive, 405
linearization error, 389–395
linearized, 372–379
measurement model, 170
measurement noise, 172
measurement update, 172
memory requirements, 478–484
mismodeling, 430–432, 447–450
monitoring, 433–434
nonconvergence, 435
observational update, 172
parameter estimation, 440–441, 455
performance, 507, 509
plant model, 170
plant noise, 170
prefiltering, 457–460, 465–466
scaling, 368
schematic, 38, 290
sequential Monte Carlo, 26
sigma-point, 26
sigmaRho, 330–346, 416
stability, 460–461
suboptimal, 436–437, 461–479
throughput, 467, 484–485
transformed state variables, 223
unobservability, 439–445
unscented, 26, 405–416
wordlength, 447, 461, 478, 480
Kalman gain, 28, 172, 176, 185
from maximum likelihod, 176
from orthogonality principle, 172
from recursive LMS, 185
Kalman–Bucy filter, 197, 199, 204
Kaminski, Paul G., 22
KFvsSKF.m, 599
Khinchin, Aleksandr Ya., 14
Klobuchar, John A., 514, 531
Klobuchar model, 514–515, 519, 531
Kolmogorov, Andrei N., 6, 14–15, 30, 75
Korean Air Lines, 512
Kronecker delta function, 118
Kronecker, Leopold, 118
Kushner, Harold J., 25, 27
Kutta, Wilhelm M., 59
Lainiotis, Demetrios G., 6
Lambert, Heinrich J., 8
Langevin, Paul, 113
Langevin equation, 113
Laning, J. Halcombe Jr., 16
Laplace, Pierre Marquis de, 6, 14
Laplace transform, 56–58
inverse, 56–57
laser gyroscope, 547
Laub, Alan J., xiii
least mean squares, 87–91
least squares, 6, 8, 87
Cholesky method, 92
computing error, 92
gramian, 9
normal equation, 9
LeastSquaresFit.m, 594
Lee, R. C. K., 20
Lefebvre, Tine, 387
Lefschetz, Solomon, 16
Legendre, Andrien-Marie, 6, 8, 14
Leibniz, Gottfried Wilhelm, 40
LGMLE, xvi, 176
likelihood function, 176
joint, 182
linear functional, 79–81, 103–104
linear LMS estimator, 171
linearization, 372–379
errors, 389–395
multi-local, 396
Riccati equation, 372
state transition, 373
LLSME, 1
LMS, 171
LMSE, xvi, 87–91
locally level coordinates, 535, 547
log-likelihood, 177
loss function, 202
LQ, xvi, 170
LRKF, 405
LSB, 335

Mahalanabis, A. K., 256, 258
Makemson, Maud W., 7
MAP, xvi
marginal benefit, 496, 499–500
marginal optimization, 496–501
marginal risk, 497–499
Markov process, 119

Markov sequence, 119
Markov, Andrei A., 6, 14, 120
Marsh, Kenneth, 418
Martin, Edward H., xiii
Maskeline, Nevil, 513
Maslow, Abraham H., 367
Math Works, 594
mathematical uncertainty, 74
MATLAB, 594
  toolboxes, 601
matrix
  characteristic values, 221
  coupling, 49–50
  covariance, 82–83
  decomposition
    Cholesky, 21, 405
    eigenvector-eigenvalue, 390
    QR, 23, 233
    singular value, 23, 390, 403
    symmetric-antisymmetric, 23
  dynamic coefficient, 49
  elementary, 306
  exponential, 56–57
  factorization, 23–24, 294–318
    Cholesky, 294–301
    in-place, 24
    triangularization, 309–318
    UD, 300–306
  fundamental solution, 52–57
  Gramian, 9–13
  Hamiltonian, 206
  information, 10, 63
  input coupling, 48–49
  Jacobian, 372–373
  measurement sensitivity, 28–29, 50, 172
  observability, 61–64
  Riccati equation, 171, 191, 204–224
  square root, 22
  state transition, 22, 28–29, 51–58
    approximation, 373
  symmetric, 298, 404
  Toeplitz, 52
  trace, 88–89, 102–104, 497–501, 517
  triangular, 21–24
    unit, 24
  triangularization, 24
  upper triangular, 52
    strictly, 52
    unit, 52
matrix fraction, 205
max, xvi
maximum likelihood, 176
Maxwell, James C., 6, 14, 119
McReynolds, Stephen R., 22, 276
mean, 82, 91
meansqesterr.m, 91, 595
measure theory, 75–77
measurement, 169
  covariance, 28–29, 183
  decorrelation, 301, 304
  noise, 338
    time-correlated, 201
  optimum, 495–501
  residual, 343
  sensitivity matrix, 28–29, 50, 172, 183
  vector, 50
Meditch, James S., 276
MediumAccuracyINS.m, 572, 600
memory requirements, 478–484
MEMS, 505, 546, 550
Mendel, Jerry M., 6
MERU, 555
metric, 93
Metropolis, Nicholas, 25, 400
mHz, xvi
mi, xvi
min, xvi
Mirelli, Vincent, 26
misalignment
  INS, 561
  sensor, 567
mismodeling, 430–432
missile
  ballistic, 505, 546
  cruise, 546
MIT, 15–16, 18, 21, 546, 555, 565
  Instrumentation Laboratory, 18, 21
ML, xvi
MLE, xvi
MMSE, 87–91
model truncation error, 63
MODL5Params.m, 600
MODL6Params.m, 601
moment, 81, 91
  central, 82
  first, 82, 91
  Gaussian, 83–85

notation, 82
raw, 81–85
recursive estimators, 91
zeroth, 82
Monte Carlo, 25, 389, 399
analysis, 399–400
sequential, 26, 402
Moore, John B., 171, 249–250, 256, 258
Moore-Penrose generalized inverse, 181, 431
Morf, Martin, 22
mph, xvi
Mueller, Fritz K., 548
multipath, 510, 517–518

NASA, 18, 20, 22
Ames Research Center, 18
JPL, 22
nautical mile, 507
navigation, 503–505
computer, 549
coordinates, 547, 551, 553, 559, 587
cruise, 565
free inertial, 565
GNSS, 506
inertial, 544
Newtonian, 545
reference point, 549
satellite, 512
Beidou, 512
Compass, 512
Doppler-based, 512
Galileo, 512
GLONASS, 512–515
GPS, 506, 512–515, 531, 533–534
history, 505–506
Timation, 513
timing-based, 513, 514
Transit, 511–513, 546, 578
solution, 534–540, 551
GNSS, 514, 534–535
INS, 551, 553
Nease, Robert F., xiii
NED coordinates, xvi, 543, 558
Newton, Isaac, 6, 40, 42
Newtonian mechanics, 42–44
Newton's laws, 37, 42
NMi, xvi, 507
noise
correlated, 200
jerk, 525–526
sensor, 180, 227
nonhomogeneous equation, 45–46, 49, 55
NonLin.m, 599
nonlinear approximation error, 389–394
nonlinear estimation, 367–420
nonlinear surveillance, 418–419
initialization, 394–398
nonlinearity, 370
norm, H_∞, 24
normal equation, 9
North-East-Down coordinates, 543
nuisance variable, 471–478

observability, 13, 61–64
obsup.m, 596
Ogata, Katsuhiko, 62
Oksendal, Bernt, 112
optimal estimator, 6, 88
optimum measurements, 495–501
orthogonality principle, 153
outcome, 74–77

Padé approximation, 57
Park, Poo Gyeon, 276
particle filter, 26, 401–402
Pascal, Blaise, 6, 14
pChiSquared.m, 595
PDOP, 517
Piazzi, Giuseppe, 7
PIGA, 548
Pinson, John C., xiii
Planck, Max, 26
plant model, 47
pLaplace.m, 595
pLogNormal.m, 595
PlotpYArctanaX.m, 595
pNormal.m, 595
Podkorytov, Andrey, xiii
point process, 27, 418
Potter, James E., 6, 21
potter.m, 598
ppm, xvi
prediction, 170, 200, 240
prefiltering, 457–460
probability, 73
Cauchy, 81–82
chi-squared, 90–91, 104, 429–430, 432

probability (*Continued*)
cumulative, 79
density, 74, 77–79
transformation, 100–102
affine, 95–96
linear, 94–95
nonlinear, 96–101
foundations, 73
Gaussian, 77–79
integral, 76–77
measure, 76–77
moment, 81, 91
space, 76–77
ProbCond.m, 596
PSD, xvi, 15, 122
pseudorandom, 120
sequence, 120–121
vector, 120
Gaussian, 120
pseudorange, 180–181, 392, 506, 518–520
differences, 542
error model, 518–520
pUniform.m, 595
pYArctanaX.m, 595

Q (disturbance noise covariance)
Q_k vs $Q(t)$, 146, 150
units, 146
Q-factor, 145
QR decomposition, 23
quadratic loss function, 202
quasi-linear, 25, 39, 370

$R(t)$ vs R_k, 152
radar tracking, 225
Ragazzini, John R., 16, 131
random process, 111–112, 114
autoregressive, 131
ergodic, 119
exponentially correlated, 126
Gaussian, 120
linear, 124
Markov, 119
mean, 115
mean power, 124
model, 113–114, 124–133
normal, 120
orthogonal, 118
stationary, 119
strict-sense, 119
weak sense, 119
wide-sense, 119
statistical properties, 114
uncorrelated, 118
white noise, 118
zero-mean, 116
random sequence, 111, 114, 127
exponentially correlated, 129
i.i.d., 114–115
Markov, 119
models, 130
random variable, 77
notation, 77
realization, 77
random walk, 128, 130
rank 1 matrix, 296
modification, 296, 319, 326, 349, 360
Rauch, Herbert E., 255–256
Rauch–Tung–Striebel smoother, 255
raw moment, 81–85
reachability, 65
ReadYUMAdata.m, 534, 601
realization, 77
reconstructability, 65
RIAS, 16
Riccati, Jacopo F., 6, 17, 171
Riccati equation, 17, 20, 39–40, 435–436
algebraic, 202–223
solution, 214, 220–223
continuous, 199
differential, 204
discrete, 190
doubling method, 488–491
linearized, 372–373
partitioning, 473
transformed state variables, 223
Richardson, John M., xiii, 27, 418
Riemann, G. F. B., 114
Riemann integral, 27, 75
risk, 203, 496–499
marginal, 497–499
Rissanen, Jorma, xiii
Roll-Pitch-Yaw coordinates, 559
rotation sensor, 547
rotational dynamics, 43
roundoff error, 63, 283–285, 434, 455
sensor, 432–433
unit roundoff, 284

RP, xvi, 111–112, 114
  multi-dimensional, 120
RPY coordinates, xvi, 559
RS, xvi, 111, 114, 127
RTS smoother, 255
RTSvsKF.m, 256, 596
Runge, Karl D. T., 59
RV, xvi, 77

Sagnac effect, 547
sample-and-propagate, 398
sampling, 398–403, 406–407
  democratic, 400–401
  for nonlinearity analysis, 391
  particle, 402
  random, 400
  sequential Monte Carlo, 402
  sigma point, 402–404
  unscented transform, 406–407
satellite navigation, 512–513
  dilution of precision, 516–517
  Doppler-based, 512–513
  error modeling, 518–532
  history, 506, 510–512
  multipath effect, 510, 517–518
  pseudorange, 180–181, 392, 506, 518–520
  timing-based, 513
SatelliteGeometry.m, 533, 537, 601
scale factor, 555, 557, 567–570
Schmidt, Stanley F., 6, 18, 20, 25, 374, 471
Schmidt–Kalman filter, 374, 471, 540–542
  complexity, 477
  gain, 475
  implementation, 477
SchmidtKalmanTest.m, 541, 601
Schmidt.m, 598
Schuler, Max, 564
Schweppe, Fred C, 110, 423
Schweppe likelihood ratio detection, 105, 395
Selective Availability, 515
sensor
  acceleration, 548
  bias, 152
  noise, 180
  rotation, 547
separability, 65
sequential Monte Carlo, 26, 30
SF, xvi, 131
Shannon, Claude E., 131
shaping filter, xvi, 131–135
  whitening, 227
shootout.m, 597
sigma-point, 26, 402–404
sigmaRho filter, 330–346, 416
  continuous time, 334–339
  discrete time, 339–342
  dynamics, 334–342
  integration, 337–338
  measurement update, 343–345
  scaling, 338–342, 346
  state variables, 334
  time update, 342
signal-to-noise ratio, 244
singular value decomposition, 23, 180–181, 403
SKF, xvi, 374
SKF_Obs.m, 599
SKF_Temp.m, 599
Smith, Joseph, xiii
smoothing, 239
  Anderson-Chirarattananon bound, 244
  fixed-interval, 241, 244–256
  fixed-lag, 241–242, 256–268
    Biswas-Mahalanabis, 258–264
  fixed-point, 241–242, 268–274
  improvement over filtering, 243–244
  Rauch–Tung–Striebel, 255
  three-pass, 252–255
  two-pass, 255
SNR, 244
SoC, system-on-chip, 332
Sorenson, Harold W., 6, 191
SPDinIP.m, 598
SPDinv.m, 598
specific force, 548
SPKF, xvi
Sputnik, 511
square-root filter, 21
stable element, 549
standard deviation, 333–334
  improvement ratio, 345–346
state space, 17–18, 40–46
  notation, 29
state transition matrix, 22, 28, 29, 51–55
  properties, 53
state variable, 30, 40

state vector, 28–29, 39, 46
  augmentation, 131
  companion form, 46, 49–50
  notation, 29
  transforming, 223
stationarity, 119–120
  strict-sense, 119
  wide-sense, 119
STM, xvi, 51, 53–55
stochastic
  calculus, 27, 125
  differential equation, 27, 113, 125, 136
    steady-state solution, 140
  integral, 27
  variable, 77
strapdown INS, 550–553, 557
Stratonovich, Ruslan L., 6, 17, 25–27, 112, 114
Striebel, Charlotte T., 255
suboptimal filter, 436, 461–471
surveillance, 27, 394–398, 418–419
  initialization, 394–398
SVD, xvi, 23, 180, 299, 403
Swerling, Peter, 6, 17
symmetric product, 297
syntonization, 530–531
system identification, 26

TDOP, 517
Thiele, Thorvald N., 6, 17, 113
Thornton, Catherine L., 22, 293, 324, 326
thornton.m, 598
throughput requirements, 478, 484–485
Tietz, Johann, 7
time, 46
  continuous/discrete, 39, 47
time-invariant systems, 47, 56, 60
  controllability, 65
  observability, 62
timeup.m, 596
time-varying systems, 45, 47, 59
Toeplitz matrix, 52
TOMS, 602
trace, 88–89, 102–104, 497–501, 517
tracking, 395
  filter, 522
  problem, 5, 27, 370
Transit satellite, 511–513, 546, 578
triangularization, 23–24, 295–296, 309–318
  Gentleman, 296, 310
  Givens, 296, 310–315
  Householder, 296, 315–318
Tung, K., 255
Turner, Robert H., 22
type 2 tracker, 524–525, 527, 537
  modified, 525

UDdecomp.m, 598
UDinv.m, 598
UDInvInPlace.m, 598
udu.m, 598
Uhlmann, Jeffrey K., xiii, 26, 405
UKF, xvi, 404–416
  data flow, 413
  vs EKF, 411–416
  sampling, 406–407
  weighting, 407–408
Ulam, Stanislaw M., 25, 399
unbiased estimate, 171
uncertainty, 74, 111–112, 140
  covariance, 116–118
  modeling, 13
underdamped harmonic oscillator, 54, 123
uniform distribution, 90–91, 432–433
University of Calgary, 602
unobservability, 61, 439–445
unobservable dynamic system, 61
unscented Kalman filter, 26, 404–416
  data flow, 413
  implementation, 408–411
unscented transform, 26, 40, 405–408
  sample size, 406–407
  sampling, 406–407
  weighting, 407–408
UT, xvii, 26, 40, 405–408
utchol.m, 599
UTsimplex.m, 599
UTsimplexDemo.m, 599
UTscaled.m, 599
UTscaledDemo.m, 599

Van Dooren, Paul, 20
Van Loan, Charles F., 150
VanLoan.m, 595
variate, 77
  transformation, 93–101

affine, 95
linear, 94
nonlinear, 97
VDOP, 517
vector
characteristic, 207, 218
function, 80
Verhaegen, Michel H., 20
VerifySchmidt.m, 598
von Neumann, John, 26, 119, 400

WAAS, 515
Weatherwax, John L., xiii
Weiffenbach, George, 10
white noise, 114, 118, 128
Wiberg, Donald F., xiii, 25
Wiener, Norbert, 6, 14–15, 119
Wiener filter, 201–202
Wiener process, 114
Wiener-Hopf equation, 17
Wiener-Kolmogorov filter, 8, 15, 17, 20
Wiley website, ix, 593
wordlength, 447, 461, 478, 480
WSS, xvii, 119
WWII, 546

YUMAdata.m, 534, 601

Zadeh, Lotfi A., 16, 131
z-transform, 60