

STATE ESTIMATION FOR ROBOTICS

Timothy D. Barfoot

Copyright © 2021

*Cambridge University Press is the Official Publisher
This Unofficial Version Compiled on May 6, 2021
Send errata to <tim.barfoot@utoronto.ca>*

Revision History

- 13 May 2017 Version best matching published first edition
12 Aug 2017 Equation (4.47a): Σ_x changed to Σ_{xx}
12 Aug 2017 Page 111, bullet 4: Σ_y changed to Σ_{yy}
12 Aug 2017 Page 117, bullet (ii): changed 4(a) to 3
12 Aug 2017 Equation (4.87): $\hat{\mathbf{P}}^-$ changed to $\check{\mathbf{P}}_k$
12 Aug 2017 Equation (4.89): $\hat{\mathbf{P}}_k$ changed to $\check{\mathbf{P}}_k$
12 Aug 2017 Equation (4.102d): $\mathbf{x}_{op,k,0}$ changed to $\mathbf{x}_{op,k,i}$
12 Aug 2017 Equation (7.102): removed negative sign
22 Nov 2017 Fixed typo in Jacobi identity (page 218)
10 Dec 2017 Equation (2.52): $\Sigma_{yy}^{-1}\Sigma_{yx}$ changed to $\Sigma_{xy}\Sigma_{yy}^{-1}$
10 Dec 2017 Inline above (8.2): \mathbf{r}_i^{vi} changed to $\mathbf{r}_i^{v_k i}$
10 Dec 2017 Equation (6.26): $\mathbf{0}$ changed to $\mathbf{0}^T$
10 Dec 2017 Inline below (4.132): $\mathbf{e}(\mathbf{x}_{op}) = \mathbf{L}\mathbf{u}(\mathbf{x}_{op})$ changed to
$$\mathbf{e}(\mathbf{x}_{op}) = \mathbf{L}^{-1}\mathbf{u}(\mathbf{x}_{op})$$

10 Dec 2017 Angular acceleration: ω_{21}° changed to ω_{21}
10 Dec 2017 Equation (4.31): corrected a double comma
10 Dec 2017 Equation (4.92a): $\mathbf{n}_{k,j}$ changed to $\dot{\mathbf{y}}_{k,j}$
5 Jan 2018 Page 227, definition of $\text{Ad}(SE(3))$: removed extra
comma before final bracket
5 Jan 2018 Equation (9.55): changed ζ_j^* to ζ_j^*
5 Jan 2018 Equation (8.47): added missing \mathbf{V}^T in an intermediate
step
5 Jan 2018 Equation (4.207): changed \star to $*$
5 Jan 2018 Equation (4.165a): changed subscript ℓ to j
5 Jan 2018 Equation (4.166): changed $\mathcal{G}_{k\ell}$ to \mathcal{G}_{jk}
16 Feb 2018 Pages 267-8: corrected \mathbf{J}_ℓ^{-1} and \mathbf{J}_r^{-1} to \mathbf{J}_ℓ and \mathbf{J}_r
25 Mar 2018 Page 108: corrected ‘mean of output is’ to be ‘mean
of input is’
25 Mar 2018 Equation (6.129): corrected \mathbf{r}_b^{ba} to be \mathbf{r}_b^{ab}
13 Apr 2018 Equation (8.105): corrected $j - 1$ to $j = 1$
22 Apr 2018 Equation (2.20): added missing -1 in definition of η
12 May 2018 Equations (7.234) and (7.239): fixed two negative
signs each to repair proof
12 May 2018 Identity page: fixed summation on series definitions of
 \mathbf{T} , \mathcal{T} , and their inverses to start at $n = 0$ not $n = 1$
27 May 2018 Started an appendix with new material since first edi-
tion
13 Jul 2018 Equation (7.150): changed $\mathbf{T}_{21} = \mathbf{T}_1\mathbf{T}_2^{-1}$ to $\mathbf{T}_{21} =$
 $\mathbf{T}_2\mathbf{T}_1^{-1}$ just below this equation in the text

13 Jul 2018	Equation (3.99b): made second term negative
13 Aug 2018	Equation (10.18): corrected J_u to J_v
13 Aug 2018	Exercise 6.6.6: adjusted wording to due to poor use of the term ‘affine’
22 Nov 2018	Equation (4.85): corrected \mathbf{n}'_i to be $\mathbf{n}'_{k,i}$
26 Nov 2018	Exercise 2.5.6: corrected N to K in upper limit of mean summation
28 Nov 2018	Added Appendix A.3 on the posterior covariance recursion for the Cholesky and RTS smoothers.
31 Jan 2019	Page 29, cleaned up an inconsistency involving M
24 Feb 2019	Equation (3.225): corrected upper integration limit from Δt_k to $\Delta t_{k:k-1}$
29 Apr 2019	Equations (3.97), (3.98), (3.101), (3.102): adjustments made to fix an inconsistent dimension problem
23 Jul 2019	Equation (4.77): $\mathbf{v}_{1:k-1}$ corrected to $\mathbf{v}_{1:k}$
1 Oct 2019	Equation (4.151) and (4.152): \mathbf{v} corrected to $\boldsymbol{\nu}$
21 Oct 2019	Equations (4.171), (4.177b), (4.178), Exercise 4.6.5: $\mathbf{G}_k(\mathbf{x})$ corrected to $\mathbf{G}_k(\mathbf{x})^T$
25 Jan 2020	Below (7.197) in text: $\phi \in \mathbb{R}$ corrected to \mathbb{R}^3
17 Apr 2020	Problem 3.6.2: corrected the size of the matrix for $\mathbf{H}^T \mathbf{W}^{-1} \mathbf{H}$
26 Apr 2020	Equation (4.30): removed an extra comma in the first term
8 May 2020	Figures 5.1, 5.2, 5.3 (and Exercises 5.5.1 and 5.5.2): adjusted time subscripts to match system matrices in main body
8 May 2020	Equation (5.3): added missing transpose symbols to each term
8 May 2020	Equation (5.11): added missing term in the underbrace of first term in second line
8 May 2020	Section 7.1.7: renamed variables to be more clear
6 Jun 2020	Identity page: fixed error in $(\mathbf{v}^\wedge \mathbf{u}^\wedge \mathbf{v})^\wedge$ identity
6 Jun 2020	Equations (7.108) and (7.119): fixed errors in coefficients of expansions
3 Jul 2020	Example 8.1: added a missing 1/6 in multiple places for weight
3 Jul 2020	Page 316: corrected negative signs in definition of \mathcal{M}
21 Oct 2020	Equation (6.99): removed negative sign in front of $\boldsymbol{\nu}$ to be easier to compare to next chapter
5 Jan 2021	Example 8.1: corrected minimal cost to be $J = 4$
5 Jan 2021	Equation (6.57c): corrected sign of second term
6 May 2021	Equations (7.227b, 7.258b, 7.260b, 7.261): corrected missing negative sign and transpose

Contents

<i>Acronyms and Abbreviations</i>	xi
<i>Notation</i>	xiii
<i>Foreword</i>	xv
1 Introduction	1
1.1 A Little History	1
1.2 Sensors, Measurements, and Problem Definition	3
1.3 How This Book Is Organized	4
1.4 Relationship to Other Books	5
Part I Estimation Machinery	7
2 Primer on Probability Theory	9
2.1 Probability Density Functions	9
2.1.1 Definitions	9
2.1.2 Bayes' Rule and Inference	10
2.1.3 Moments	11
2.1.4 Sample Mean and Covariance	12
2.1.5 Statistically Independent, Uncorrelated	12
2.1.6 Normalized Product	13
2.1.7 Shannon and Mutual Information	14
2.1.8 Cramér-Rao Lower Bound and Fisher Information	14
2.2 Gaussian Probability Density Functions	15
2.2.1 Definitions	15
2.2.2 Isserlis' Theorem	16
2.2.3 Joint Gaussian PDFs, Their Factors, and Inference	18
2.2.4 Statistically Independent, Uncorrelated	20
2.2.5 Linear Change of Variables	20
2.2.6 Normalized Product of Gaussians	22
2.2.7 Sherman-Morrison-Woodbury Identity	23
2.2.8 Passing a Gaussian through a Nonlinearity	24
2.2.9 Shannon Information of a Gaussian	28
2.2.10 Mutual Information of a Joint Gaussian PDF	30
2.2.11 Cramér-Rao Lower Bound Applied to Gaussian PDFs	30
2.3 Gaussian Processes	32
2.4 Summary	33
2.5 Exercises	33

3	Linear-Gaussian Estimation	37
3.1	Batch Discrete-Time Estimation	37
3.1.1	Problem Setup	37
3.1.2	Maximum A Posteriori	39
3.1.3	Bayesian Inference	44
3.1.4	Existence, Uniqueness, and Observability	46
3.1.5	MAP Covariance	50
3.2	Recursive Discrete-Time Smoothing	51
3.2.1	Exploiting Sparsity in the Batch Solution	52
3.2.2	Cholesky Smoother	53
3.2.3	Rauch-Tung-Striebel Smoother	55
3.3	Recursive Discrete-Time Filtering	58
3.3.1	Factoring the Batch Solution	59
3.3.2	Kalman Filter via MAP	63
3.3.3	Kalman Filter via Bayesian Inference	68
3.3.4	Kalman Filter via Gain Optimization	69
3.3.5	Kalman Filter Discussion	70
3.3.6	Error Dynamics	71
3.3.7	Existence, Uniqueness, and Observability	72
3.4	Batch Continuous-Time Estimation	74
3.4.1	Gaussian Process Regression	74
3.4.2	A Class of Exactly Sparse Gaussian Process Priors	77
3.4.3	Linear Time-Invariant Case	83
3.4.4	Relationship to Batch Discrete-Time Estimation	87
3.5	Summary	88
3.6	Exercises	88
4	Nonlinear Non-Gaussian Estimation	91
4.1	Introduction	91
4.1.1	Full Bayesian Estimation	92
4.1.2	Maximum a Posteriori Estimation	94
4.2	Recursive Discrete-Time Estimation	96
4.2.1	Problem Setup	96
4.2.2	Bayes Filter	97
4.2.3	Extended Kalman Filter	100
4.2.4	Generalized Gaussian Filter	103
4.2.5	Iterated Extended Kalman Filter	105
4.2.6	IEKF Is a MAP Estimator	106
4.2.7	Alternatives for Passing PDFs through Nonlinearities	107
4.2.8	Particle Filter	116
4.2.9	Sigmapoint Kalman Filter	118
4.2.10	Iterated Sigmapoint Kalman Filter	123
4.2.11	ISPKF Seeks the Posterior Mean	126
4.2.12	Taxonomy of Filters	127
4.3	Batch Discrete-Time Estimation	127
4.3.1	Maximum A Posteriori	128
4.3.2	Bayesian Inference	135
4.3.3	Maximum Likelihood	137
4.3.4	Discussion	142

<i>Contents</i>	vii
-----------------	-----

4.4	Batch Continuous-Time Estimation	143
4.4.1	Motion Model	143
4.4.2	Observation Model	146
4.4.3	Bayesian Inference	146
4.4.4	Algorithm Summary	147
4.5	Summary	148
4.6	Exercises	149
5	Biases, Correspondences, and Outliers	151
5.1	Handling Input/Measurement Biases	152
5.1.1	Bias Effects on the Kalman Filter	152
5.1.2	Unknown Input Bias	155
5.1.3	Unknown Measurement Bias	157
5.2	Data Association	159
5.2.1	External Data Association	160
5.2.2	Internal Data Association	160
5.3	Handling Outliers	161
5.3.1	RANSAC	162
5.3.2	M-Estimation	163
5.3.3	Covariance Estimation	166
5.4	Summary	168
5.5	Exercises	168
Part II Three-Dimensional Machinery		171
6	Primer on Three-Dimensional Geometry	173
6.1	Vectors and Reference Frames	173
6.1.1	Reference Frames	174
6.1.2	Dot Product	174
6.1.3	Cross Product	175
6.2	Rotations	176
6.2.1	Rotation Matrices	176
6.2.2	Principal Rotations	177
6.2.3	Alternate Rotation Representations	178
6.2.4	Rotational Kinematics	184
6.2.5	Perturbing Rotations	188
6.3	Poses	192
6.3.1	Transformation Matrices	193
6.3.2	Robotics Conventions	194
6.3.3	Frenet-Serret Frame	196
6.4	Sensor Models	199
6.4.1	Perspective Camera	199
6.4.2	Stereo Camera	206
6.4.3	Range-Azimuth-Elevation	208
6.4.4	Inertial Measurement Unit	209
6.5	Summary	211
6.6	Exercises	212

7	Matrix Lie Groups	215
7.1	Geometry	215
7.1.1	Special Orthogonal and Special Euclidean Groups	215
7.1.2	Lie Algebras	217
7.1.3	Exponential Map	219
7.1.4	Adjoint	226
7.1.5	Baker-Campbell-Hausdorff	230
7.1.6	Distance, Volume, Integration	237
7.1.7	Interpolation	240
7.1.8	Homogeneous Points	246
7.1.9	Calculus and Optimization	246
7.1.10	Identities	254
7.2	Kinematics	255
7.2.1	Rotations	255
7.2.2	Poses	258
7.2.3	Linearized Rotations	261
7.2.4	Linearized Poses	265
7.3	Probability and Statistics	266
7.3.1	Gaussian Random Variables and PDFs	267
7.3.2	Uncertainty on a Rotated Vector	271
7.3.3	Compounding Poses	273
7.3.4	Fusing Poses	280
7.3.5	Propagating Uncertainty through a Nonlinear Camera Model	285
7.4	Summary	292
7.5	Exercises	293
Part III Applications		295
8	Pose Estimation Problems	297
8.1	Point-Cloud Alignment	297
8.1.1	Problem Setup	298
8.1.2	Unit-Length Quaternion Solution	298
8.1.3	Rotation Matrix Solution	302
8.1.4	Transformation Matrix Solution	316
8.2	Point-Cloud Tracking	319
8.2.1	Problem Setup	319
8.2.2	Motion Priors	320
8.2.3	Measurement Model	321
8.2.4	EKF Solution	322
8.2.5	Batch Maximum a Posteriori Solution	325
8.3	Pose-Graph Relaxation	329
8.3.1	Problem Setup	329
8.3.2	Batch Maximum Likelihood Solution	330
8.3.3	Initialization	333
8.3.4	Exploiting Sparsity	333
8.3.5	Chain Example	334

<i>Contents</i>	ix
9 Pose-and-Point Estimation Problems	337
9.1 Bundle Adjustment	337
9.1.1 Problem Setup	338
9.1.2 Measurement Model	338
9.1.3 Maximum Likelihood Solution	342
9.1.4 Exploiting Sparsity	345
9.1.5 Interpolation Example	348
9.2 Simultaneous Localization and Mapping	352
9.2.1 Problem Setup	352
9.2.2 Batch Maximum a Posteriori Solution	353
9.2.3 Exploiting Sparsity	354
9.2.4 Example	355
10 Continuous-Time Estimation	357
10.1 Motion Prior	357
10.1.1 General	357
10.1.2 Simplification	361
10.2 Simultaneous Trajectory Estimation and Mapping	362
10.2.1 Problem Setup	363
10.2.2 Measurement Model	363
10.2.3 Batch Maximum a Posteriori Solution	364
10.2.4 Exploiting Sparsity	365
10.2.5 Interpolation	366
10.2.6 Postscript	367
<i>References</i>	369
<i>Index</i>	375
Appendix A Supplementary Material	379
A.1 Lie Group Tools	379
A.1.1 SE(3) Derivative	379
A.2 Kinematics	380
A.2.1 SO(3) Jacobian Identity	380
A.2.2 SE(3) Jacobian Identity	380
A.3 Smoothers	381
A.3.1 Posterior Covariance in the Cholesky Smoother	381
A.3.2 Posterior Covariance in the RTS Smoother	383

Acronyms and Abbreviations

BA	bundle adjustment	337
BCH	Baker-Campbell-Hausdorff	231
BLUE	best linear unbiased estimate	70
CRLB	Cramér-Rao lower bound	14
DARCES	data-aligned rigidity-constrained exhaustive search	161
EKF	extended Kalman filter	70
GP	Gaussian process	32
GPS	Global Positioning System	4
ICP	iterative closest point	297
IEKF	iterated extended Kalman filter	105
IMU	inertial measurement unit	209
IRLS	iteratively reweighted least squares	165
ISPKF	iterated sigmapoint Kalman filter	123
KF	Kalman filter	37
LDU	lower-diagonal-upper	23
LG	linear-Gaussian	38
LTI	linear time-invariant	83
LTV	linear time-varying	77
MAP	maximum a posteriori	4
ML	maximum likelihood	138
NASA	National Aeronautics and Space Administration	3
NLNG	nonlinear, non-Gaussian	91
PDF	probability density function	9
RAE	range-azimuth-elevation	208
RANSAC	random sample consensus	162
RTS	Rauch-Tung-Striebel	55
SDE	stochastic differential equation	77
SLAM	simultaneous localization and mapping	158
SMW	Sherman-Morrison-Woodbury	23
SP	sigmapoint	110
SPKF	sigmapoint Kalman filter	118
STEAM	simultaneous trajectory estimation and mapping	362
SWF	sliding-window filter	142
UDL	upper-diagonal-lower	23
UKF	unscented Kalman filter (also called SPKF)	119

Notation

– GENERAL NOTATION –

a	This font is used for quantities that are real scalars
\mathbf{a}	This font is used for quantities that are real column vectors
\mathbf{A}	This font is used for quantities that are real matrices
\mathbf{A}	This font is used for time-invariant system quantities
$p(\mathbf{a})$	The probability density of \mathbf{a}
$p(\mathbf{a} \mathbf{b})$	The probability density of \mathbf{a} given \mathbf{b}
$\mathcal{N}(\mathbf{a}, \mathbf{B})$	Gaussian probability density with mean \mathbf{a} and covariance \mathbf{B}
$\mathcal{GP}(\boldsymbol{\mu}(t), \mathcal{K}(t, t'))$	Gaussian process with mean function, $\boldsymbol{\mu}(t)$, and covariance function, $\mathcal{K}(t, t')$
\mathcal{O}	Observability matrix
$(\cdot)_k$	The value of a quantity at timestep k
$(\cdot)_{k_1:k_2}$	The set of values of a quantity from timestep k_1 to timestep k_2 , inclusive
$\overrightarrow{\mathcal{F}}_a$	A vectrix representing a reference frame in three dimensions
$\overset{a}{\overrightarrow{\wedge}}$	A vector quantity in three dimensions
$(\cdot)^{\times}$	The cross-product operator, which produces a skew-symmetric matrix from a 3×1 column
$\mathbf{1}$	The identity matrix
$\mathbf{0}$	The zero matrix
$\mathbb{R}^{M \times N}$	The vectorspace of real $M \times N$ matrices
$(\hat{\cdot})$	A posterior (estimated) quantity
$(\check{\cdot})$	A prior quantity

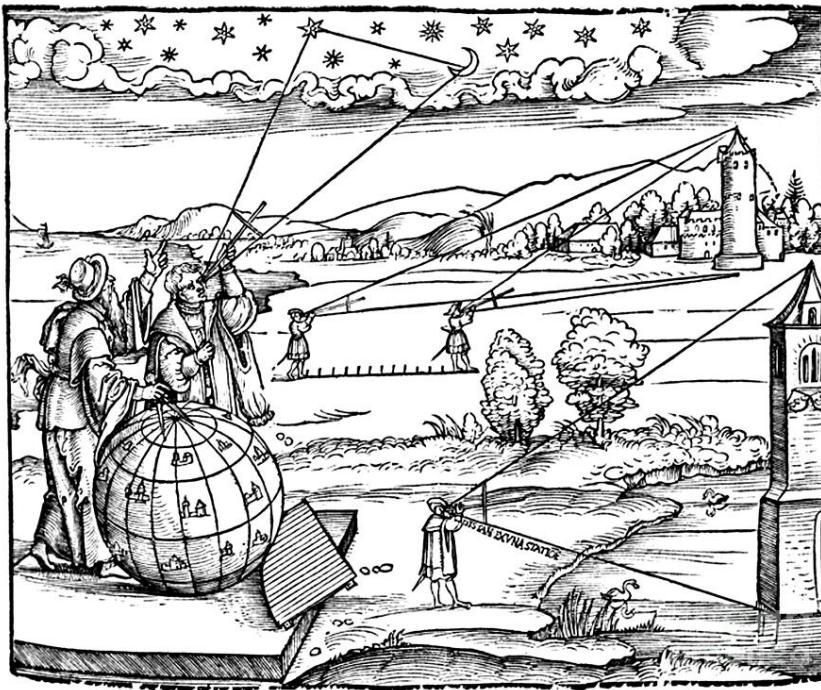
– MATRIX-LIE-GROUP NOTATION –

$SO(3)$	The special orthogonal group, a matrix Lie group used to represent rotations
$\mathfrak{so}(3)$	The Lie algebra associated with $SO(3)$
$SE(3)$	The special Euclidean group, a matrix Lie group used to represent poses
$\mathfrak{se}(3)$	The Lie algebra associated with $SE(3)$
$(\cdot)^\wedge$	An operator associated with the Lie algebra for rotations and poses
$(\cdot)^\lambda$	An operator associated with the adjoint of an element from the Lie algebra for poses
$\text{Ad}(\cdot)$	An operator producing the adjoint of an element from the Lie group for rotations and poses
$\text{ad}(\cdot)$	An operator producing the adjoint of an element from the Lie algebra for rotations and poses
\mathbf{C}_{ba}	A 3×3 rotation matrix (member of $SO(3)$) that takes points expressed in $\underline{\mathcal{F}}_a$ and re-expresses them in $\underline{\mathcal{F}}_b$, which is rotated with respect to $\underline{\mathcal{F}}_a$
\mathbf{T}_{ba}	A 4×4 transformation matrix (member of $SE(3)$) that takes points expressed in $\underline{\mathcal{F}}_a$ and re-expresses them in $\underline{\mathcal{F}}_b$, which is rotated/translated with respect to $\underline{\mathcal{F}}_a$
\mathcal{T}_{ba}	A 6×6 adjoint of a transformation matrix (member of $\text{Ad}(SE(3))$)

Foreword

My interest in state estimation stems from the field of mobile robotics, particularly for space exploration. Within mobile robotics, there has been an explosion of research referred to as *probabilistic robotics*. With computing resources becoming very inexpensive, and the advent of rich new sensing technologies, such as digital cameras and laser rangefinders, robotics has been at the forefront of developing exciting new ideas in the area of state estimation.

In particular, this field was probably the first to find practical applications of the so-called Bayes filter, a much more general technique than the famous Kalman filter. In just the last few years, mobile robotics has even started going beyond the Bayes filter to batch, nonlinear optimization-based techniques, with very promising results. Because my primary area of interest is navigation of robots in outdoor environ-



*Introductio
Geographica* by
Petrus Apianus
(1495-1552), a
German
mathematician,
astronomer, and
cartographer.
Much of
three-dimensional
state estimation
has to do with
triangulation
and/or
trilateration; we
measure some
angles and lengths
and infer the
others through
trigonometry.

ments, I have often been faced with vehicles operating in three dimensions. Accordingly, I have attempted to provide a detailed look at how to approach state estimation in three dimensions. In particular, I show how to treat rotations and poses in a simple and practical way using matrix Lie groups. The reader should have a background in undergraduate linear algebra and calculus, but otherwise, this book is fairly standalone. I hope readers of these pages will find something useful; I know I learned a great deal while creating them.

I have provided some historical notes in the margins throughout the book, mostly in the form of biographical sketches of some of the researchers after whom various concepts and techniques are named; I primarily used Wikipedia as the source for this information. Also, the first part of Chapter 6 (up to the alternate rotation parameterizations), which introduces three-dimensional geometry, is based heavily on notes originally produced by Chris Damaren at the University of Toronto Institute for Aerospace Studies.

This book would not have been possible without the collaborations of many fantastic graduate students along the way. Paul Furgale's PhD thesis extended my understanding of matrix Lie groups significantly by introducing me to their use for describing poses; this led us on an interesting journey into the details of transformation matrices and how to use them effectively in estimation problems. Paul's later work led me to become interested in continuous-time estimation. Chi Hay Tong's PhD thesis introduced me to the use of Gaussian processes in estimation theory, and he helped immensely in working out the details of the continuous-time methods presented herein; my knowledge in this area was further improved through collaborations with Simo Särkkä from Aalto University while on sabbatical at the University of Oxford. Additionally, I learned a great deal by working with Sean Anderson, Patrick Carle, Hang Dong, Andrew Lambert, Keith Leung, Colin McManus, and Braden Stenning; each of their projects added to my understanding of state estimation. Colin, in particular, encouraged me several times to turn my notes from my graduate course on state estimation into this book.

I am indebted to Gabriele D'Eleuterio, who set me on the path of studying rotations and reference frames in the context of dynamics; many of the tools he showed me transferred effortlessly to state estimation. He also taught me the importance of clean, unambiguous notation.

Finally, thanks to all those who read and pointed out errors in the drafts of this book, particularly Marc Gallant and Shu-Hua Tsao, who found many typos, and James Forbes, who volunteered to read and provide comments.

1

Introduction

Robotics inherently deals with things that move in the world. We live in an era of rovers on Mars, drones surveying the Earth, and, soon, self-driving cars. And, although specific robots have their subtleties, there are also some common issues we must face in all applications, particularly *state estimation* and *control*.

The *state* of a robot is a set of quantities, such as position, orientation, and velocity, that, if known, fully describe that robot's motion over time. Here we focus entirely on the problem of estimating the state of a robot, putting aside the notion of control. Yes, control is essential, as we would like to make our robots behave in a certain way. But, the first step in doing so is often the process of determining the state. Moreover, the difficulty of state estimation is often underestimated for real-world problems, and thus it is important to put it on an equal footing with control.

In this book, we introduce the classic estimation results for linear systems corrupted by Gaussian measurement noise. We then examine some of the extensions to nonlinear systems with non-Gaussian noise. In a departure from typical estimation texts, we take a detailed look at how to tailor general estimation results to robots operating in three-dimensional space, advocating a particular approach to handling rotations.

The rest of this introduction provides a little history of estimation, discusses types of sensors and measurements, and introduces the problem of state estimation. It concludes with a breakdown of the contents of the book and provides some other suggested reading.

1.1 A Little History

About 4,000 years ago, the early seafarers were faced with a vehicular state estimation problem: how to determine a ship's position while at sea. Early attempts to develop primitive charts and make observations of the sun allowed local navigation along coastlines. However, it was not until the fifteenth century that global navigation on the open sea became possible with the advent of key technologies and tools. The mariner's compass, an early form of the magnetic compass, allowed

Figure 1.1
Quadrant. A tool used to measure angles.

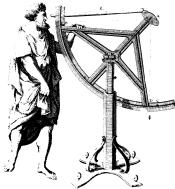


Figure 1.2
Harrison's H4. The first clock able to keep accurate time at sea, enabling determination of longitude.



Carl Friedrich Gauss (1777-1855) was a German mathematician who contributed significantly to many fields including statistics and estimation.

Rudolf Emil Kálmán (1930-2016) was a Hungarian-born American electrical engineer, mathematician, and inventor.

crude measurements of direction to be made. Together with coarse nautical charts, the compass made it possible to sail along rhumb lines between key destinations (i.e., following a compass bearing). A series of instruments was then gradually invented that made it possible to measure the angle between distant points (i.e., cross-staff, astrolabe, quadrant, sextant, theodolite) with increasing accuracy.

These instruments allowed latitude to be determined at sea fairly readily using celestial navigation. For example, in the Northern Hemisphere, the angle between the North Star, Polaris, and the horizon provides the latitude. Longitude, however, was a much more difficult problem. It was known early on that an accurate timepiece was the missing piece of the puzzle for the determination of longitude. The behaviours of key celestial bodies appear differently at different locations on the Earth. Knowing the time of day therefore allows longitude to be inferred. In 1764, British clockmaker John Harrison built the first accurate portable timepiece that effectively solved the longitude problem; a ship's longitude could be determined to within about 10 nautical miles.

Estimation theory also finds its roots in astronomy. The method of least squares was pioneered¹ by Gauss, who developed the technique to minimize the impact of measurement error in the prediction of orbits. Gauss reportedly used least squares to predict the position of the dwarf planet Ceres after passing behind the Sun, accurate to within half a degree (about nine months after it was last seen). The year was 1801, and Gauss was 23. Later, in 1809, he proved that the least-squares method is optimal under the assumption of normally distributed errors. Most of the classic estimation techniques in use today can be directly related to Gauss' least-squares method.

The idea of fitting models to minimize the impact of measurement error carried forward, but it was not until the middle of the twentieth century that estimation really took off. This was likely correlated with the dawn of the computer age. In 1960, Kalman published two landmark papers that have defined much of what has followed in the field of state estimation. First, he introduced the notion of *observability* (Kalman, 1960a), which tells us when a state can be inferred from a set of measurements in a dynamic system. Second, he introduced an optimal framework for estimating a system's state in the presence of measurement noise (Kalman, 1960b); this classic technique for linear systems (whose measurements are corrupted by Gaussian noise) is famously known as the *Kalman filter*, and has been the workhorse of estimation for the more than 50 years since its inception. Although used in many fields, it has been widely adopted in aerospace appli-

¹ There is some debate as to whether Adrien Marie Legendre might have come up with least squares before Gauss.

cations. Researchers at the *National Aeronautics and Space Administration (NASA)* were the first to employ the Kalman filter to aid in the estimation of spacecraft trajectories on the Ranger, Mariner, and Apollo programs. In particular, the on-board computer on the Apollo 11 lunar module, the first manned spacecraft to land on the surface of the Moon, employed a Kalman filter to estimate the module's position above the lunar surface based on noisy radar measurements.

Many incremental improvements have been made to the field of state estimation since these early milestones. Faster and cheaper computers have allowed much more computationally complex techniques to be implemented in practical systems. However, until about 15 years ago, it seemed that estimation was possibly waning as an active research area. But, something has happened to change that; exciting new sensing technologies are coming along (e.g., digital cameras, laser imaging, the Global Positioning System) that pose new challenges to this old field.

1.2 Sensors, Measurements, and Problem Definition

To understand the need for state estimation is to understand the nature of sensors. All sensors have a limited precision. Therefore, all measurements derived from real sensors have associated uncertainty. Some sensors are better at measuring specific quantities than others, but even the best sensors still have a degree of imprecision. When we combine various sensor measurements into a state estimate, it is important to keep track of all the uncertainties involved and therefore it is hoped know how confident we can be in our estimate.

In a way, state estimation is about doing the best we can with the sensors we have. This, however, does not prevent us from, in parallel, improving the quality of our sensors. A good example is the *theodolite* sensor that was developed in 1787 to allow triangulation across the English Channel. It was much more precise than its predecessors and helped show that much of England was poorly mapped by tying measurements to well-mapped France.

It is useful to put sensors into two categories: *interoceptive*² and *exteroceptive*. These are actually terms borrowed from human physiology, but they have become somewhat common in engineering. Some definitions follow³:

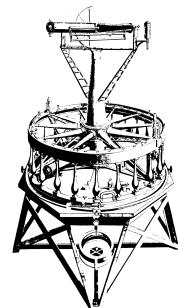
in·tero·cep·tive [int-ə-rō-'sep-tiv], *adjective*: of, relating to, or being stimuli arising within the body.

ex·tero·cep·tive [ek-stə-rō-'sep-tiv], *adjective*: relating to, being, or activated by stimuli received by an organism from outside.

EARLY ESTIMATION MILESTONES

1654	Pascal and Fermat lay foundations of probability theory
1764	Bayes' rule
1801	Gauss uses least-squares to estimate the orbit of the planetoid Ceres
1805	Legendre publishes "least-squares"
1913	Markov chains
1933	(Chapman)-Kolmogorov equations
1949	Wiener filter
1960	Kalman (Bucy) filter
1965	Rauch-Tung-Striebel smoother
1970	Jazwinski coins "Bayes filter"

Figure 1.3
Theodolite. A better tool to measure angles.



² Sometimes *proprioceptive* is used synonymously.

³ Merriam-Webster's Dictionary.

Typical interoceptive sensors are the accelerometer (measures translational acceleration), gyroscope (measures angular rate), and wheel odometer (measures angular rate). Typical exteroceptive sensors are the camera (measures range/bearing to a landmark or landmarks) and time-of-flight transmitter/receiver (e.g., laser rangefinder, pseudolites, *Global Positioning System (GPS)* transmitter/receiver). Roughly speaking, we can think of exteroceptive measurements as being of the position and orientation of a vehicle, whereas interoceptive ones are of a vehicle's velocity or acceleration. In most cases, the best state estimation concepts make use of both interoceptive and exteroceptive measurements. For example, the combination of a GPS receiver (exteroceptive) and an inertial measurement unit (three linear accelerometers and three rate gyros; interoceptive) is a popular means of estimating a vehicle's position/velocity on Earth. And, the combination of a Sun/star sensor (exteroceptive) and three rate gyros (interoceptive) is commonly used to carry out pose determination on satellites.

Now that we understand a little bit about sensors, we are prepared to define the problem that will be investigated in this book:

Estimation is the problem of reconstructing the underlying state of a system given a sequence of measurements as well as a prior model of the system.

There are many specific versions of this problem and just as many solutions. The goal is to understand which methods work well in which situations, in order to pick the best tool for the job.

1.3 How This Book Is Organized

The book is broken into three main parts:

- I: Estimation Machinery
- II: Three-Dimensional Machinery
- III: Applications

The first part, *Estimation Machinery*, presents classic and state-of-the-art estimation tools, without the complication of dealing with things that live in three-dimensional space (and therefore translate and rotate); the state to be estimated is assumed to be a generic vector. For those not interested in the details of working in three-dimensional space, this first part can be read in a standalone manner. It covers both recursive state estimation techniques and batch methods (less common in classic estimation books). As is commonplace in robotics and machine learning today, we adopt a *Bayesian* approach to estimation in this book. We contrast (full) Bayesian methods with *maximum a posteriori (MAP)* methods, and attempt to make clear the difference between these when faced with nonlinear problems. The book also connects continuous-time estimation with Gaussian process regression

from the machine-learning world. Finally, it touches on some practical issues, such as robust estimation and biases.

The second part, *Three-Dimensional Machinery*, provides a basic primer on three-dimensional geometry and gives a detailed but accessible introduction to matrix Lie groups. To represent an object in three-dimensional space, we need to talk about that object's translation and rotation. The rotational part turns out to be a problem for our estimation tools because rotations are not *vectors* in the usual sense and so we cannot naively apply the methods from Part I to three-dimensional robotics problems involving rotations. Part II, therefore, examines the geometry, kinematics, and probability/statistics of rotations and poses (translation plus rotation).

Finally, in the third part, *Applications*, the first two parts of the book are brought together. We look at a number of classic three-dimensional estimation problems involving objects translating and rotating in three-dimensional space. We show how to adapt the methods from Part I based on the knowledge gained in Part II. The result is a suite of easy-to-implement methods for three-dimensional state estimation. The spirit of these examples can also, we hope, be adapted to create other novel techniques moving forward.

1.4 Relationship to Other Books

There are many other good books on state estimation and robotics, but very few cover both topics simultaneously. We briefly describe a few recent works that do cover these topics and their relationships to this book.

Probabilistic Robotics by Thrun et al. (2006) is a great introduction to mobile robotics, with a large focus on state estimation in relation to mapping and localization. It covers the probabilistic paradigm that is dominant in much of robotics today. It mainly describes robots operating in the two-dimensional, horizontal plane. The probabilistic methods described are not necessarily limited to the two-dimensional case, but the details of extending to three dimensions are not provided.

Computational Principles of Mobile Robotics by Dudek and Jenkin (2010) is a great overview book on mobile robotics that touches on state estimation, again in relation to localization and mapping methods. It does not work out the details of performing state estimation in three dimensions.

Mobile Robotics: Mathematics, Models, and Methods by Kelly (2013) is another excellent book on mobile robotics and covers state estimation extensively. Three-dimensional situations are covered, particularly in relation to satellite-based and inertial navigation. As the book covers all aspects of robotics, it does not delve deeply into how to handle rotational variables within three-dimensional state estimation.

Robotics, Vision, and Control by Corke (2011) is another great and comprehensive book that covers state estimation for robotics, including in three dimensions. Similarly to the previously mentioned book, the breadth of Corke’s book necessitates that it not delve too deeply into the specific aspects of state estimation treated herein.

Bayesian Filtering and Smoothing by Särkkä (2013) is a super book focused on recursive Bayesian methods. It covers the recursive methods in far more depth than this book, but does not cover batch methods nor focus on the details of carrying out estimation in three dimensions.

Stochastic Models, Information Theory, and Lie Groups: Classical Results and Geometric Methods by Chirikjian (2009), an excellent two-volume work, is perhaps the closest in content to the current book. It explicitly investigates the consequences of carrying out state estimation on matrix Lie groups (and hence rotational variables). It is quite theoretical in nature and goes beyond the current book in this sense, covering applications beyond robotics.

Engineering Applications of Noncommutative Harmonic Analysis: With Emphasis on Rotation and Motion Groups by Chirikjian and Kyatkin (2001) and the recent update, *Harmonic Analysis for Engineers and Applied Scientists: Updated and Expanded Edition* (Chirikjian and Kyatkin, 2016), also provide key insights to representing probability globally on Lie groups. In the current book, we limit ourselves to approximate methods that are appropriate to the situation where rotational uncertainty is not too high.

Although it is not an estimation book per se, it is worth mentioning *Optimization on Matrix Manifolds* by Absil et al. (2009), which provides a detailed look at how to handle optimization problems when the quantity being optimized is not necessarily a vector, a concept that is quite relevant to robotics because rotations do not behave like vectors (they form a Lie group).

The current book is somewhat unique in focusing only on state estimation and working out the details of common three-dimensional robotics problems in enough detail to be easily implemented for many practical situations.

Part I

Estimation Machinery

2

Primer on Probability Theory

In what follows, we will be using a number of basic concepts from probability and statistics. This chapter serves to provide a review of these concepts. For a classic book on probability and random processes, see Papoulis (1965). For a light read on the history of probability theory, Devlin (2008) provides a wonderful introduction; this book also helps to understand the difference between the *frequentist* and *Bayesian* views of probability. We will primarily adopt the latter in our approach to estimation, although this chapter mentions some basic frequentist statistical concepts in passing. We begin by discussing general *probability density functions (PDFs)* and then focus on the specific case of Gaussian PDFs. The chapter concludes by introducing Gaussian processes, the continuous-time version of Gaussian random variables.

2.1 Probability Density Functions

2.1.1 Definitions

We say that a *random variable*, x , is distributed according to a particular PDF. Let $p(x)$ be a PDF for the random variable, x , over the interval $[a, b]$. This is a non-negative function that satisfies

$$\int_a^b p(x) dx = 1. \quad (2.1)$$

That is, it satisfies the *axiom of total probability*. Note that this is *probability density*, not *probability*.

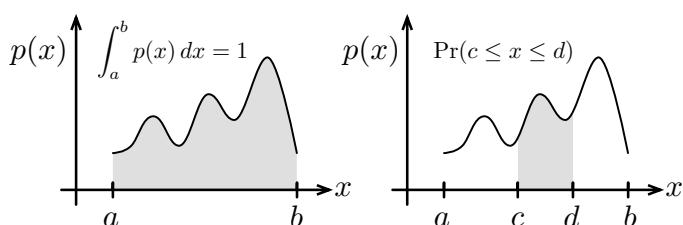


Figure 2.1
Probability density over a finite interval (left). Probability of being within a sub-interval (right).

Probability is given by the area under the density function¹. For example, the probability that x lies between c and d , $\Pr(c \leq x \leq d)$, is given by

$$\Pr(c \leq x \leq d) = \int_c^d p(x) dx. \quad (2.2)$$

Figure 2.1 depicts a general PDF over a finite interval as well as the probability of being within a sub-interval. We will use PDFs to represent the *likelihood* of x being in all possible states in the interval, $[a, b]$, given some evidence in the form of data.

We can also introduce a conditioning variable. Let $p(x|y)$ be a PDF over $x \in [a, b]$ conditioned on $y \in [r, s]$ such that

$$(\forall y) \quad \int_a^b p(x|y) dx = 1. \quad (2.3)$$

We may also denote *joint probability densities* for N -dimensional continuous variables in our framework as $p(\mathbf{x})$, where $\mathbf{x} = (x_1, \dots, x_N)$ with $x_i \in [a_i, b_i]$. Note that we can also use the notation

$$p(x_1, x_2, \dots, x_N) \quad (2.4)$$

in place of $p(\mathbf{x})$. Sometimes we even mix and match the two and write

$$p(\mathbf{x}, \mathbf{y}) \quad (2.5)$$

for the joint density of \mathbf{x} and \mathbf{y} . In the N -dimensional case, the axiom of total probability requires

$$\int_{\mathbf{a}}^{\mathbf{b}} p(\mathbf{x}) d\mathbf{x} = \int_{a_N}^{b_N} \cdots \int_{a_2}^{b_2} \int_{a_1}^{b_1} p(x_1, x_2, \dots, x_N) dx_1 dx_2 \cdots dx_N = 1, \quad (2.6)$$

where $\mathbf{a} = (a_1, a_2, \dots, a_N)$ and $\mathbf{b} = (b_1, b_2, \dots, b_N)$. In what follows, we will sometimes simplify notation by leaving out the integration limits, \mathbf{a} and \mathbf{b} .

2.1.2 Bayes' Rule and Inference

We can always factor a joint probability density into a conditional and a non-conditional factor²:

$$p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x}|\mathbf{y})p(\mathbf{y}) = p(\mathbf{y}|\mathbf{x})p(\mathbf{x}). \quad (2.7)$$

¹ The classical treatment of probability theory starts with probability distributions, Kolmogorov's three axioms, and works out the details of probability densities as a consequence of being the derivative of probability distributions. As is common in robotics, we will work directly with densities in a Bayesian framework, and therefore we will skip these formalities and present only the results we need using densities. We shall be careful to use the term *density* not *distribution* as we are working with continuous variables throughout this book.

² In the specific case that \mathbf{x} and \mathbf{y} are *statistically independent*, we can factor the joint density as $p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x})p(\mathbf{y})$.

Rearranging gives *Bayes' rule*:

$$p(\mathbf{x}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{x})p(\mathbf{x})}{p(\mathbf{y})}. \quad (2.8)$$

We can use this to *infer* the *posterior* or likelihood of the state given the measurements, $p(\mathbf{x}|\mathbf{y})$, if we have a *prior* PDF over the state, $p(\mathbf{x})$, and the sensor model, $p(\mathbf{y}|\mathbf{x})$. We do this by expanding the denominator so that

$$p(\mathbf{x}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{x})p(\mathbf{x})}{\int p(\mathbf{y}|\mathbf{x})p(\mathbf{x}) d\mathbf{x}}. \quad (2.9)$$

We compute the denominator, $p(\mathbf{y})$, by *marginalization* as follows³:

$$\begin{aligned} p(\mathbf{y}) &= p(\mathbf{y}) \underbrace{\int p(\mathbf{x}|\mathbf{y}) d\mathbf{x}}_1 = \int p(\mathbf{x}|\mathbf{y})p(\mathbf{y}) d\mathbf{x} \\ &= \int p(\mathbf{x}, \mathbf{y}) d\mathbf{x} = \int p(\mathbf{y}|\mathbf{x})p(\mathbf{x}) d\mathbf{x}, \end{aligned} \quad (2.10)$$

which can be quite expensive to do in the general nonlinear case.

Note that in Bayesian inference, $p(\mathbf{x})$ is known as the *prior* density, while $p(\mathbf{x}|\mathbf{y})$ is known as the *posterior* density. Thus, all *a priori* information is encapsulated in $p(\mathbf{x})$, while $p(\mathbf{x}|\mathbf{y})$ contains the *a posteriori* information.

2.1.3 Moments

When working with mass distributions (a.k.a., density functions) in dynamics, we often keep track of only a few properties called the *moments* of mass (e.g., mass, center of mass, inertia matrix). The same is true with PDFs. The zeroth probability moment is always 1 since this is exactly the axiom of total probability. The first probability moment is known as the *mean*, μ :

$$\mu = E[\mathbf{x}] = \int \mathbf{x} p(\mathbf{x}) d\mathbf{x}, \quad (2.11)$$

where $E[\cdot]$ denotes the expectation operator. For a general matrix function, $\mathbf{F}(\mathbf{x})$, the expectation is written as

$$E[\mathbf{F}(\mathbf{x})] = \int \mathbf{F}(\mathbf{x}) p(\mathbf{x}) d\mathbf{x}, \quad (2.12)$$

but note that we must interpret this as

$$E[\mathbf{F}(\mathbf{x})] = [E[f_{ij}(\mathbf{x})]] = [\int f_{ij}(\mathbf{x}) p(\mathbf{x}) d\mathbf{x}]. \quad (2.13)$$

Thomas Bayes (1701-1761) was an English statistician, philosopher, and Presbyterian minister, known for having formulated a specific case of the theorem that bears his name. Bayes never published what would eventually become his most famous accomplishment; his notes were edited and published after his death by Richard Price (Bayes, 1764).

³ When integration limits are not stated, they are assumed to be over the entire allowable domain of the variable; e.g., \mathbf{x} from \mathbf{a} to \mathbf{b} .

The second probability moment is known as the *covariance matrix*, Σ :

$$\Sigma = E [(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^T]. \quad (2.14)$$

The next two moments are called the *skewness* and *kurtosis*, but for the multivariate case these get quite complicated and require tensor representations. We will not need them here, but it should be mentioned that there are an infinite number of these probability moments.

2.1.4 Sample Mean and Covariance

Suppose we have a random variable, \mathbf{x} , and an associated PDF, $p(\mathbf{x})$. We can draw samples from this density, which we denote as:

$$\mathbf{x}_{\text{meas}} \leftarrow p(\mathbf{x}). \quad (2.15)$$

A sample is sometimes referred to as a *realization* of the random variable, and we can think of it intuitively as a measurement. If we drew N such samples and wanted to estimate the mean and covariance of random variable, \mathbf{x} , we could use the *sample mean* and *sample covariance* to do so:

$$\boldsymbol{\mu}_{\text{meas}} = \frac{1}{N} \sum_{i=1}^N \mathbf{x}_{i,\text{meas}}, \quad (2.16a)$$

$$\Sigma_{\text{meas}} = \frac{1}{N-1} \sum_{i=1}^N (\mathbf{x}_{i,\text{meas}} - \boldsymbol{\mu}_{\text{meas}}) (\mathbf{x}_{i,\text{meas}} - \boldsymbol{\mu}_{\text{meas}})^T. \quad (2.16b)$$

Friedrich Wilhelm Bessel (1784-1846) was a German astronomer, mathematician (systematizer of the Bessel functions, which were discovered by Bernoulli). He was the first astronomer to determine the distance from the sun to another star by the method of parallax. The Bessel correction is technically a factor of $N/(N-1)$ that is multiplied in front of the ‘biased’ formula for covariance that divides by N instead of $N-1$.

Notably, the normalization in the sample covariance uses $N-1$ rather than N in the denominator, which is referred to as *Bessel’s correction*. Intuitively, this is necessary because the sample covariance uses the difference of the measurements with the sample mean, which itself is computed from the same measurements, resulting in a slight correlation. The sample covariance can be shown to be an unbiased estimate of the true covariance, and it is also ‘larger’ than when N is used in the denominator. It is also worth mentioning that as N becomes large, $N-1 \approx N$, so the bias effect for which sample covariance compensates becomes less pronounced.

2.1.5 Statistically Independent, Uncorrelated

If we have two random variables, \mathbf{x} and \mathbf{y} , we say that the variables are *statistically independent* if their joint density factors as follows:

$$p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x}) p(\mathbf{y}). \quad (2.17)$$

We say that the variables are *uncorrelated* if

$$E [\mathbf{x}\mathbf{y}^T] = E [\mathbf{x}] E [\mathbf{y}]^T. \quad (2.18)$$

If the variables are statistically independent, this implies they are also uncorrelated. However, the reverse is not true in general for all types of densities⁴. We will often exploit (or assume) that variables are statistically independent to simplify computations.

2.1.6 Normalized Product

An operation that it sometimes useful is to take the *normalized product* of two PDFs over the same variable⁵. If $p_1(\mathbf{x})$ and $p_2(\mathbf{x})$ are two PDFs for \mathbf{x} , the normalized product, $p(\mathbf{x})$, is formed as

$$p(\mathbf{x}) = \eta p_1(\mathbf{x}) p_2(\mathbf{x}), \quad (2.19)$$

where

$$\eta = \left(\int p_1(\mathbf{x}) p_2(\mathbf{x}) d\mathbf{x} \right)^{-1}, \quad (2.20)$$

is a normalization constant to ensure $p(\mathbf{x})$ satisfies the axiom of total probability.

In a Bayesian context, the normalized product can be used to fuse independent estimates of a variable (represented as PDFs) under the assumption of a uniform prior:

$$p(\mathbf{x}|\mathbf{y}_1, \mathbf{y}_2) = \eta p(\mathbf{x}|\mathbf{y}_1) p(\mathbf{x}|\mathbf{y}_2), \quad (2.21)$$

where η is again a normalization constant to enforce the axiom of total probability. To see this, we begin by writing the left-hand side using Bayes' rule:

$$p(\mathbf{x}|\mathbf{y}_1, \mathbf{y}_2) = \frac{p(\mathbf{y}_1, \mathbf{y}_2|\mathbf{x})p(\mathbf{x})}{p(\mathbf{y}_1, \mathbf{y}_2)}. \quad (2.22)$$

Assuming statistical independence of \mathbf{y}_1 and \mathbf{y}_2 given \mathbf{x} (e.g., measurements corrupted by statistically independent noise), we have

$$p(\mathbf{y}_1, \mathbf{y}_2|\mathbf{x}) = p(\mathbf{y}_1|\mathbf{x}) p(\mathbf{y}_2|\mathbf{x}) = \frac{p(\mathbf{x}|\mathbf{y}_1)p(\mathbf{y}_1)}{p(\mathbf{x})} \frac{p(\mathbf{x}|\mathbf{y}_2)p(\mathbf{y}_2)}{p(\mathbf{x})}, \quad (2.23)$$

where we have used Bayes' rule once again on the individual factors. Substituting this into (2.22), we have

$$p(\mathbf{x}|\mathbf{y}_1, \mathbf{y}_2) = \eta p(\mathbf{x}|\mathbf{y}_1) p(\mathbf{x}|\mathbf{y}_2), \quad (2.24)$$

where

$$\eta = \frac{p(\mathbf{y}_1)p(\mathbf{y}_2)}{p(\mathbf{y}_1, \mathbf{y}_2)p(\mathbf{x})}. \quad (2.25)$$

If we let the prior, $p(\mathbf{x})$, be uniform over all values of \mathbf{x} (i.e., constant), then η is also a constant and (2.24) is an instance of the normalized product described above.

⁴ It is true for Gaussian PDFs, as discussed below.

⁵ This is quite different than when we are working with a joint density over two variables.

Claude Elwood Shannon (1916-2001) was an American mathematician, electronic engineer, and cryptographer known as ‘the father of information theory’ (Shannon, 1948).

Harald Cramér (1893-1985) was a Swedish mathematician, actuary, and statistician, specializing in mathematical statistics and probabilistic number theory. Calyampudi Radhakrishna Rao, (1920-present) is an Indian American mathematician and statistician. Cramér and Rao were amongst the first to derive what is now known as the CRLB.

2.1.7 Shannon and Mutual Information

Often we have estimated a PDF for some random variable and then want to quantify how certain we are in, for example, the mean of that PDF. One method of doing this is to look at the *negative entropy* or *Shannon information*, H , which is given by

$$H(\mathbf{x}) = -E[\ln p(\mathbf{x})] = - \int p(\mathbf{x}) \ln p(\mathbf{x}) d\mathbf{x}. \quad (2.26)$$

We will make this expression specific to Gaussian PDFs below.

Another useful quantity is the *mutual information*, $I(\mathbf{x}, \mathbf{y})$, between two random variables, \mathbf{x} and \mathbf{y} , given by

$$I(\mathbf{x}, \mathbf{y}) = E \left[\ln \left(\frac{p(\mathbf{x}, \mathbf{y})}{p(\mathbf{x})p(\mathbf{y})} \right) \right] = \iint p(\mathbf{x}, \mathbf{y}) \ln \left(\frac{p(\mathbf{x}, \mathbf{y})}{p(\mathbf{x})p(\mathbf{y})} \right) d\mathbf{x} d\mathbf{y}. \quad (2.27)$$

Mutual information measures how much knowing one of the variables reduces uncertainty about the other. When \mathbf{x} and \mathbf{y} are statistically independent, we have

$$\begin{aligned} I(\mathbf{x}, \mathbf{y}) &= \iint p(\mathbf{x}) p(\mathbf{y}) \ln \left(\frac{p(\mathbf{x})p(\mathbf{y})}{p(\mathbf{x})p(\mathbf{y})} \right) d\mathbf{x} d\mathbf{y} \\ &= \iint p(\mathbf{x}) p(\mathbf{y}) \underbrace{\ln(1)}_0 d\mathbf{x} d\mathbf{y} = 0. \end{aligned} \quad (2.28)$$

When \mathbf{x} and \mathbf{y} are dependent, we have $I(\mathbf{x}, \mathbf{y}) \geq 0$. We also have the useful relationship

$$I(\mathbf{x}, \mathbf{y}) = H(\mathbf{x}) + H(\mathbf{y}) - H(\mathbf{x}, \mathbf{y}), \quad (2.29)$$

relating mutual information and Shannon information.

2.1.8 Cramér-Rao Lower Bound and Fisher Information

Suppose we have a deterministic parameter, $\boldsymbol{\theta}$, that influences the outcome of a random variable, \mathbf{x} . This can be captured by writing the PDF for \mathbf{x} as depending on $\boldsymbol{\theta}$:

$$p(\mathbf{x}|\boldsymbol{\theta}). \quad (2.30)$$

Furthermore, suppose we now draw a sample, \mathbf{x}_{meas} , from $p(\mathbf{x}|\boldsymbol{\theta})$:

$$\mathbf{x}_{\text{meas}} \leftarrow p(\mathbf{x}|\boldsymbol{\theta}). \quad (2.31)$$

The \mathbf{x}_{meas} is sometimes called a *realization* of the random variable \mathbf{x} ; we can think of it as a ‘measurement’⁶.

Then, the *Cramér-Rao lower bound (CRLB)* says that the covariance of any *unbiased estimate*⁷, $\hat{\boldsymbol{\theta}}$ (based on the measurement, \mathbf{x}_{meas}), of

⁶ We use the subscript, ‘meas’, to indicate it is a measurement.

⁷ We will use $(\hat{\cdot})$ to indicate an *estimated* quantity.

the deterministic parameter, $\boldsymbol{\theta}$, is bounded by the *Fisher information matrix*, $\mathbf{I}(\mathbf{x}|\boldsymbol{\theta})$:

$$\text{cov}(\hat{\boldsymbol{\theta}}|\mathbf{x}_{\text{meas}}) = E \left[(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta})(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta})^T \right] \geq \mathbf{I}^{-1}(\mathbf{x}|\boldsymbol{\theta}), \quad (2.32)$$

where ‘unbiased’ implies $E[\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}] = \mathbf{0}$ and ‘bounded’ means

$$\text{cov}(\hat{\boldsymbol{\theta}}|\mathbf{x}_{\text{meas}}) - \mathbf{I}^{-1}(\mathbf{x}|\boldsymbol{\theta}) \geq 0, \quad (2.33)$$

i.e., positive-semi-definite. The Fisher information matrix is given by

$$\mathbf{I}(\mathbf{x}|\boldsymbol{\theta}) = E \left[\left(\frac{\partial \ln p(\mathbf{x}|\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \right)^T \left(\frac{\partial \ln p(\mathbf{x}|\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \right) \right]. \quad (2.34)$$

The CRLB therefore sets a fundamental limit on how certain we can be about an estimate of a parameter, given our measurements.

Sir Ronald Aylmer Fisher (1890-1962)
was an English statistician, evolutionary biologist, geneticist, and eugenicist. His contributions to statistics include the analysis of variance, method of maximum likelihood, fiducial inference, and the derivation of various sampling distributions.

2.2 Gaussian Probability Density Functions

2.2.1 Definitions

In much of what is to follow, we will be working with Gaussian PDFs. In one dimension, a Gaussian PDF is given by

$$p(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp \left(-\frac{1}{2} \frac{(x-\mu)^2}{\sigma^2} \right), \quad (2.35)$$

where μ is the *mean* and σ^2 is the *variance* (σ is called the *standard deviation*). Figure 2.2 shows a one-dimensional Gaussian PDF.

A multivariate Gaussian PDF, $p(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$, over the random variable, $\mathbf{x} \in \mathbb{R}^N$, may be expressed as

$$p(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^N \det \boldsymbol{\Sigma}}} \exp \left(-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right), \quad (2.36)$$

where $\boldsymbol{\mu} \in \mathbb{R}^N$ is the mean and $\boldsymbol{\Sigma} \in \mathbb{R}^{N \times N}$ is the (symmetric positive-definite) covariance matrix. Thus, for a Gaussian we have that

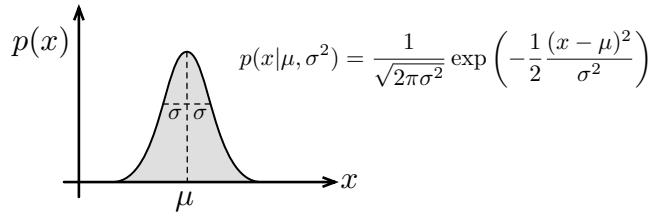
$$\boldsymbol{\mu} = E[\mathbf{x}] = \int_{-\infty}^{\infty} \mathbf{x} \frac{1}{\sqrt{(2\pi)^N \det \boldsymbol{\Sigma}}} \exp \left(-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right) d\mathbf{x}, \quad (2.37)$$

and

$$\begin{aligned} \boldsymbol{\Sigma} &= E[(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^T] \\ &= \int_{-\infty}^{\infty} (\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^T \frac{1}{\sqrt{(2\pi)^N \det \boldsymbol{\Sigma}}} \\ &\quad \times \exp \left(-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right) d\mathbf{x}. \end{aligned} \quad (2.38)$$

Figure 2.2

One-dimensional Gaussian PDF. A notable property of a Gaussian is that the mean and mode (most likely x) are both at μ .



We may also write that \mathbf{x} is *normally* (a.k.a., Gaussian) distributed using the following notation:

$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}).$$

We say a random variable is *standard normally* distributed if

$$\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{1}),$$

where $\mathbf{1}$ is an $N \times N$ identity matrix.

2.2.2 Isserlis' Theorem

Moments of multivariate Gaussian PDFs get a little messy to compute beyond the usual mean and covariance, but there are some specific cases that we will make use of later that are worth discussing. We can use *Isserlis' theorem* to compute higher-order moments of a Gaussian random variable, $\mathbf{x} = (x_1, x_2, \dots, x_{2M}) \in \mathbb{R}^{2M}$. In general, this theorem says

$$E[x_1 x_2 x_3 \cdots x_{2M}] = \sum \prod E[x_i x_j], \quad (2.39)$$

where this implies summing over all distinct ways of partitioning into a product of M pairs. This implies that there are $\frac{(2M)!}{(2^M M!)}$ terms in the sum. With four variables we have

$$E[x_i x_j x_k x_\ell] = E[x_i x_j] E[x_k x_\ell] + E[x_i x_k] E[x_j x_\ell] + E[x_i x_\ell] E[x_j x_k]. \quad (2.40)$$

We can apply this theorem to work out some useful results for matrix expressions.

Assume we have $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}) \in \mathbb{R}^N$. We will have occasion to compute expressions of the form

$$E \left[\mathbf{x} (\mathbf{x}^T \mathbf{x})^p \mathbf{x}^T \right], \quad (2.41)$$

where p is a non-negative integer. Trivially, when $p = 0$, we simply have

Leon Isserlis (1881-1966) was a Russian-born British statistician known for his work on the exact distribution of sample moments.

$E[\mathbf{x}\mathbf{x}^T] = \Sigma$. When $p = 1$, we have⁸

$$\begin{aligned}
E[\mathbf{x}\mathbf{x}^T\mathbf{x}\mathbf{x}^T] &= E\left[\left[x_i x_j \left(\sum_{k=1}^N x_k^2\right)\right]_{ij}\right] = \left[\sum_{k=1}^N E[x_i x_j x_k^2]\right]_{ij} \\
&= \left[\sum_{k=1}^N (E[x_i x_j] E[x_k^2] + 2E[x_i x_k] E[x_k x_j])\right]_{ij} \\
&= [E[x_i x_j]]_{ij} \sum_{k=1}^N E[x_k^2] + 2 \left[\sum_{k=1}^N E[x_i x_k] E[x_k x_j]\right]_{ij} \\
&= \Sigma \text{tr}(\Sigma) + 2\Sigma^2 \\
&= \Sigma (\text{tr}(\Sigma)\mathbf{1} + 2\Sigma). \tag{2.42}
\end{aligned}$$

Note that in the scalar case we have $x \sim \mathcal{N}(0, \sigma^2)$ and hence $E[x^4] = \sigma^2(\sigma^2 + 2\sigma^2) = 3\sigma^4$, a well-known result. Results for $p > 1$ are possible using a similar approach, but we do not compute them for now.

We also consider the case where

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{12}^T & \Sigma_{22} \end{bmatrix}\right), \tag{2.43}$$

with $\dim(\mathbf{x}_1) = N_1$ and $\dim(\mathbf{x}_2) = N_2$. We will need to compute expressions of the form

$$E[\mathbf{x}(\mathbf{x}_1^T \mathbf{x}_1)^p \mathbf{x}^T], \tag{2.44}$$

where p is a non-negative integer. Again, when $p = 0$, we trivially have $E[\mathbf{x}\mathbf{x}^T] = \Sigma$. When $p = 1$, we have

$$\begin{aligned}
E[\mathbf{x}\mathbf{x}_1^T \mathbf{x}_1 \mathbf{x}^T] &= E\left[\left[x_i x_j \left(\sum_{k=1}^{N_1} x_k^2\right)\right]_{ij}\right] = \left[\sum_{k=1}^{N_1} E[x_i x_j x_k^2]\right]_{ij} \\
&= \left[\sum_{k=1}^{N_1} (E[x_i x_j] E[x_k^2] + 2E[x_i x_k] E[x_k x_j])\right]_{ij} \\
&= [E[x_i x_j]]_{ij} \sum_{k=1}^{N_1} E[x_k^2] + 2 \left[\sum_{k=1}^{N_1} E[x_i x_k] E[x_k x_j]\right]_{ij} \\
&= \Sigma \text{tr}(\Sigma_{11}) + 2 \begin{bmatrix} \Sigma_{11}^2 & \Sigma_{11} \Sigma_{12} \\ \Sigma_{12}^T \Sigma_{11} & \Sigma_{12}^T \Sigma_{12} \end{bmatrix} \\
&= \Sigma \left(\text{tr}(\Sigma_{11})\mathbf{1} + 2 \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \right). \tag{2.45}
\end{aligned}$$

⁸ The notation $[\cdot]_{ij}$ implies populating the matrix $\mathbf{A} = [a_{ij}]$ with the appropriate ij th entry in each element.

Similarly, we have

$$\begin{aligned} E[\mathbf{x}\mathbf{x}_2^T\mathbf{x}_2\mathbf{x}^T] &= \boldsymbol{\Sigma} \operatorname{tr}(\boldsymbol{\Sigma}_{22}) + 2 \begin{bmatrix} \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{12}^T & \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22} \\ \boldsymbol{\Sigma}_{22}\boldsymbol{\Sigma}_{12}^T & \boldsymbol{\Sigma}_{22}^2 \end{bmatrix} \\ &= \boldsymbol{\Sigma} \left(\operatorname{tr}(\boldsymbol{\Sigma}_{22})\mathbf{1} + 2 \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \boldsymbol{\Sigma}_{12}^T & \boldsymbol{\Sigma}_{22} \end{bmatrix} \right), \end{aligned} \quad (2.46)$$

and as a final check,

$$E[\mathbf{x}\mathbf{x}^T\mathbf{x}\mathbf{x}^T] = E[\mathbf{x}(\mathbf{x}_1^T\mathbf{x}_1 + \mathbf{x}_2^T\mathbf{x}_2)\mathbf{x}^T] = E[\mathbf{x}\mathbf{x}_1^T\mathbf{x}_1\mathbf{x}^T] + E[\mathbf{x}\mathbf{x}_2^T\mathbf{x}_2\mathbf{x}^T]. \quad (2.47)$$

We furthermore have that

$$\begin{aligned} E[\mathbf{x}\mathbf{x}^T\mathbf{A}\mathbf{x}\mathbf{x}^T] &= E \left[\left[x_i x_j \left(\sum_{k=1}^N \sum_{\ell=1}^N x_k a_{k\ell} x_\ell \right) \right]_{ij} \right] \\ &= \left[\sum_{k=1}^N \sum_{\ell=1}^N a_{k\ell} E[x_i x_j x_k x_\ell] \right]_{ij} \\ &= \left[\sum_{k=1}^N \sum_{\ell=1}^N a_{k\ell} (E[x_i x_j] E[x_k x_\ell] + E[x_i x_k] E[x_j x_\ell] \right. \\ &\quad \left. + E[x_i x_\ell] E[x_j x_k]) \right]_{ij} \\ &= [E[x_i x_j]]_{ij} \left(\sum_{k=1}^N \sum_{\ell=1}^N a_{k\ell} E[x_k x_\ell] \right) \\ &\quad + \left[\sum_{k=1}^N \sum_{\ell=1}^N E[x_i x_k] a_{k\ell} E[x_\ell x_j] \right]_{ij} \\ &\quad + \left[\sum_{k=1}^N \sum_{\ell=1}^N E[x_i x_\ell] a_{k\ell} E[x_k x_j] \right]_{ij} \\ &= \boldsymbol{\Sigma} \operatorname{tr}(\mathbf{A}\boldsymbol{\Sigma}) + \boldsymbol{\Sigma}\mathbf{A}\boldsymbol{\Sigma} + \boldsymbol{\Sigma}\mathbf{A}^T\boldsymbol{\Sigma} \\ &= \boldsymbol{\Sigma} (\operatorname{tr}(\mathbf{A}\boldsymbol{\Sigma}) \mathbf{1} + \mathbf{A}\boldsymbol{\Sigma} + \mathbf{A}^T\boldsymbol{\Sigma}), \end{aligned} \quad (2.48)$$

where \mathbf{A} is a compatible square matrix.

2.2.3 Joint Gaussian PDFs, Their Factors, and Inference

We can also have a joint Gaussian over a pair of variables, (\mathbf{x}, \mathbf{y}) , which we write as

$$p(\mathbf{x}, \mathbf{y}) = \mathcal{N} \left(\begin{bmatrix} \boldsymbol{\mu}_x \\ \boldsymbol{\mu}_y \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{xx} & \boldsymbol{\Sigma}_{xy} \\ \boldsymbol{\Sigma}_{yx} & \boldsymbol{\Sigma}_{yy} \end{bmatrix} \right), \quad (2.49)$$

which has the same exponential form as (2.36). Note that $\boldsymbol{\Sigma}_{yx} = \boldsymbol{\Sigma}_{xy}^T$. It is always possible to break a joint density into the product of two

factors, $p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x}|\mathbf{y})p(\mathbf{y})$, and we can work out the details for the joint Gaussian case by using the *Schur complement*⁹. We begin by noting that

$$\begin{bmatrix} \Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_{yy} \end{bmatrix} = \begin{bmatrix} \mathbf{1} & \Sigma_{xy}\Sigma_{yy}^{-1} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \begin{bmatrix} \Sigma_{xx} - \Sigma_{xy}\Sigma_{yy}^{-1}\Sigma_{yx} & \mathbf{0} \\ \mathbf{0} & \Sigma_{yy} \end{bmatrix} \begin{bmatrix} \mathbf{1} & \mathbf{0} \\ \Sigma_{yy}^{-1}\Sigma_{yx} & \mathbf{1} \end{bmatrix}, \quad (2.50)$$

where $\mathbf{1}$ is the identity matrix. We then invert both sides to find that

$$\begin{aligned} \begin{bmatrix} \Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_{yy} \end{bmatrix}^{-1} &= \begin{bmatrix} \mathbf{1} & \mathbf{0} \\ -\Sigma_{yy}^{-1}\Sigma_{yx} & \mathbf{1} \end{bmatrix} \\ &\times \begin{bmatrix} (\Sigma_{xx} - \Sigma_{xy}\Sigma_{yy}^{-1}\Sigma_{yx})^{-1} & \mathbf{0} \\ \mathbf{0} & \Sigma_{yy}^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{1} & -\Sigma_{xy}\Sigma_{yy}^{-1} \\ \mathbf{0} & \mathbf{1} \end{bmatrix}. \end{aligned} \quad (2.51)$$

Looking just to the quadratic part (inside the exponential) of the joint PDF, $p(\mathbf{x}, \mathbf{y})$, we have

$$\begin{aligned} &\left(\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} - \begin{bmatrix} \boldsymbol{\mu}_x \\ \boldsymbol{\mu}_y \end{bmatrix} \right)^T \begin{bmatrix} \Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_{yy} \end{bmatrix}^{-1} \left(\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} - \begin{bmatrix} \boldsymbol{\mu}_x \\ \boldsymbol{\mu}_y \end{bmatrix} \right) \\ &= \left(\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} - \begin{bmatrix} \boldsymbol{\mu}_x \\ \boldsymbol{\mu}_y \end{bmatrix} \right)^T \begin{bmatrix} \mathbf{1} & \mathbf{0} \\ -\Sigma_{yy}^{-1}\Sigma_{yx} & \mathbf{1} \end{bmatrix} \begin{bmatrix} (\Sigma_{xx} - \Sigma_{xy}\Sigma_{yy}^{-1}\Sigma_{yx})^{-1} & \mathbf{0} \\ \mathbf{0} & \Sigma_{yy}^{-1} \end{bmatrix} \\ &\quad \times \begin{bmatrix} \mathbf{1} & -\Sigma_{xy}\Sigma_{yy}^{-1} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \left(\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} - \begin{bmatrix} \boldsymbol{\mu}_x \\ \boldsymbol{\mu}_y \end{bmatrix} \right) \\ &= (\mathbf{x} - \boldsymbol{\mu}_x - \Sigma_{xy}\Sigma_{yy}^{-1}(\mathbf{y} - \boldsymbol{\mu}_y))^T (\Sigma_{xx} - \Sigma_{xy}\Sigma_{yy}^{-1}\Sigma_{yx})^{-1} \\ &\quad \times (\mathbf{x} - \boldsymbol{\mu}_x - \Sigma_{xy}\Sigma_{yy}^{-1}(\mathbf{y} - \boldsymbol{\mu}_y)) + (\mathbf{y} - \boldsymbol{\mu}_y)^T \Sigma_{yy}^{-1} (\mathbf{y} - \boldsymbol{\mu}_y), \end{aligned} \quad (2.52)$$

which is the sum of two quadratic terms. Since the exponential of a sum is the product of two exponentials, we have that

$$p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x}|\mathbf{y})p(\mathbf{y}), \quad (2.53a)$$

$$p(\mathbf{x}|\mathbf{y}) = \mathcal{N}(\boldsymbol{\mu}_x + \Sigma_{xy}\Sigma_{yy}^{-1}(\mathbf{y} - \boldsymbol{\mu}_y), \Sigma_{xx} - \Sigma_{xy}\Sigma_{yy}^{-1}\Sigma_{yx}), \quad (2.53b)$$

$$p(\mathbf{y}) = \mathcal{N}(\boldsymbol{\mu}_y, \Sigma_{yy}). \quad (2.53c)$$

It is important to note that both factors, $p(\mathbf{x}|\mathbf{y})$ and $p(\mathbf{y})$, are Gaussian PDFs. Also, if we happen to know the value of \mathbf{y} (i.e., it is measured), we can work out the likelihood of \mathbf{x} given this value of \mathbf{y} by computing $p(\mathbf{x}|\mathbf{y})$ using (2.53b).

This is in fact the cornerstone of *Gaussian inference*: we start with a prior about our state, $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}_x, \Sigma_{xx})$, then narrow this down based on some measurements, \mathbf{y}_{meas} . In (2.53b), we see that an adjustment is made to the mean, $\boldsymbol{\mu}_x$, and the covariance, Σ_{xx} (it is made smaller).

Issai Schur
(1875-1941) was a German mathematician who worked on group representations (the subject with which he is most closely associated), but also in combinatorics and number theory and even in theoretical physics.

⁹ In this case, we have that the Schur complement of Σ_{yy} is the expression, $\Sigma_{xx} - \Sigma_{xy}\Sigma_{yy}^{-1}\Sigma_{yx}$.

2.2.4 Statistically Independent, Uncorrelated

In the case of Gaussian PDFs, statistically independent variables are also uncorrelated (true in general) and uncorrelated variables are also statistically independent (not true for all types of PDFs). We can see this fairly easily by looking at (2.53). If we assume statistical independence, $p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x})p(\mathbf{y})$ and so $p(\mathbf{x}|\mathbf{y}) = p(\mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}_x, \boldsymbol{\Sigma}_{xx})$. Looking at (2.53b), this implies

$$\boldsymbol{\Sigma}_{xy}\boldsymbol{\Sigma}_{yy}^{-1}(\mathbf{y} - \boldsymbol{\mu}_y) = \mathbf{0}, \quad (2.54a)$$

$$\boldsymbol{\Sigma}_{xy}\boldsymbol{\Sigma}_{yy}^{-1}\boldsymbol{\Sigma}_{yx} = \mathbf{0}, \quad (2.54b)$$

which further implies that $\boldsymbol{\Sigma}_{xy} = \mathbf{0}$. Since

$$\boldsymbol{\Sigma}_{xy} = E[(\mathbf{x} - \boldsymbol{\mu}_x)(\mathbf{y} - \boldsymbol{\mu}_y)^T] = E[\mathbf{x}\mathbf{y}^T] - E[\mathbf{x}]E[\mathbf{y}]^T, \quad (2.55)$$

we have the uncorrelated condition:

$$E[\mathbf{x}\mathbf{y}^T] = E[\mathbf{x}]E[\mathbf{y}]^T. \quad (2.56)$$

We can also work through the logic in the other direction by first assuming the variables are uncorrelated, which leads to $\boldsymbol{\Sigma}_{xy} = \mathbf{0}$, and finally to statistical independence. Since these conditions are equivalent, we will often use *statistically independent* and *uncorrelated* interchangeably in the context of Gaussian PDFs.

2.2.5 Linear Change of Variables

Suppose that we have a Gaussian random variable,

$$\mathbf{x} \in \mathbb{R}^N \sim \mathcal{N}(\boldsymbol{\mu}_x, \boldsymbol{\Sigma}_{xx}),$$

and that we have a second random variable, $\mathbf{y} \in \mathbb{R}^M$, related to \mathbf{x} through the linear map,

$$\mathbf{y} = \mathbf{G}\mathbf{x}, \quad (2.57)$$

where we assume that $\mathbf{G} \in \mathbb{R}^{M \times N}$ is a constant matrix. We would like to know what the statistical properties of \mathbf{y} are. One way to do this is to simply apply the expectation operator directly:

$$\boldsymbol{\mu}_y = E[\mathbf{y}] = E[\mathbf{G}\mathbf{x}] = \mathbf{G}E[\mathbf{x}] = \mathbf{G}\boldsymbol{\mu}_x, \quad (2.58a)$$

$$\begin{aligned} \boldsymbol{\Sigma}_{yy} &= E[(\mathbf{y} - \boldsymbol{\mu}_y)(\mathbf{y} - \boldsymbol{\mu}_y)^T] \\ &= \mathbf{G}E[(\mathbf{x} - \boldsymbol{\mu}_x)(\mathbf{x} - \boldsymbol{\mu}_x)^T]\mathbf{G}^T = \mathbf{G}\boldsymbol{\Sigma}_{xx}\mathbf{G}^T, \end{aligned} \quad (2.58b)$$

so that we have $\mathbf{y} \sim \mathcal{N}(\boldsymbol{\mu}_y, \boldsymbol{\Sigma}_{yy}) = \mathcal{N}(\mathbf{G}\boldsymbol{\mu}_x, \mathbf{G}\boldsymbol{\Sigma}_{xx}\mathbf{G}^T)$.

Another way to look at this is a change of variables. We assume that the linear map is *injective*, meaning two \mathbf{x} values cannot map to a single \mathbf{y} value; in fact, let us simplify the injective condition by assuming a

stricter condition, that \mathbf{G} is invertible (and hence $M = N$). The axiom of total probability lets us write,

$$\int_{-\infty}^{\infty} p(\mathbf{x}) d\mathbf{x} = 1. \quad (2.59)$$

A small volume of \mathbf{x} is related to a small volume of \mathbf{y} by

$$d\mathbf{y} = |\det \mathbf{G}| d\mathbf{x}. \quad (2.60)$$

We can then make a substitution of variables to have

$$\begin{aligned} 1 &= \int_{-\infty}^{\infty} p(\mathbf{x}) d\mathbf{x} \\ &= \int_{-\infty}^{\infty} \frac{1}{\sqrt{(2\pi)^N \det \Sigma_{xx}}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_x)^T \Sigma_{xx}^{-1} (\mathbf{x} - \boldsymbol{\mu}_x)\right) d\mathbf{x} \\ &= \int_{-\infty}^{\infty} \frac{1}{\sqrt{(2\pi)^N \det \Sigma_{xx}}} \\ &\quad \times \exp\left(-\frac{1}{2}(\mathbf{G}^{-1}\mathbf{y} - \boldsymbol{\mu}_x)^T \Sigma_{xx}^{-1} (\mathbf{G}^{-1}\mathbf{y} - \boldsymbol{\mu}_x)\right) |\det \mathbf{G}|^{-1} d\mathbf{y} \\ &= \int_{-\infty}^{\infty} \frac{1}{\sqrt{(2\pi)^N \det \mathbf{G} \det \Sigma_{xx} \det \mathbf{G}^T}} \\ &\quad \times \exp\left(-\frac{1}{2}(\mathbf{y} - \mathbf{G}\boldsymbol{\mu}_x)^T \mathbf{G}^{-T} \Sigma_{xx}^{-1} \mathbf{G}^{-1} (\mathbf{y} - \mathbf{G}\boldsymbol{\mu}_x)\right) d\mathbf{y} \\ &= \int_{-\infty}^{\infty} \frac{1}{\sqrt{(2\pi)^N \det(\mathbf{G}\Sigma_{xx}\mathbf{G}^T)}} \\ &\quad \times \exp\left(-\frac{1}{2}(\mathbf{y} - \mathbf{G}\boldsymbol{\mu}_x)^T (\mathbf{G}\Sigma_{xx}\mathbf{G}^T)^{-1} (\mathbf{y} - \mathbf{G}\boldsymbol{\mu}_x)\right) d\mathbf{y}, \end{aligned} \quad (2.61)$$

whereupon we have $\boldsymbol{\mu}_y = \mathbf{G}\boldsymbol{\mu}_x$ and $\Sigma_{yy} = \mathbf{G}\Sigma_{xx}\mathbf{G}^T$, as before. If $M < N$, our linear mapping is no longer injective and the change of variable approach cannot be used to map statistics from \mathbf{x} to \mathbf{y} .

We can also think about going in the other direction from \mathbf{y} to \mathbf{x} , assuming $M < N$ and $\text{rank } \mathbf{G} = M$. This is a bit tricky, as the resulting covariance for \mathbf{x} will blow up since we are dilating¹⁰ to a larger space. To get around this, we switch to *information form*. Letting

$$\mathbf{u} = \Sigma_{yy}^{-1}\mathbf{y}, \quad (2.62)$$

we have that

$$\mathbf{u} \sim \mathcal{N}(\Sigma_{yy}^{-1}\boldsymbol{\mu}_y, \Sigma_{yy}^{-1}). \quad (2.63)$$

Likewise, letting

$$\mathbf{v} = \Sigma_{xx}^{-1}\mathbf{x}, \quad (2.64)$$

¹⁰ *Dilation* is the opposite of *projection*.

we have that

$$\mathbf{v} \sim \mathcal{N}(\boldsymbol{\Sigma}_{xx}^{-1}\boldsymbol{\mu}_x, \boldsymbol{\Sigma}_{xx}^{-1}). \quad (2.65)$$

Since the mapping from \mathbf{y} to \mathbf{x} is not unique, we need to specify what we want to do. One choice is to let

$$\mathbf{v} = \mathbf{G}^T \mathbf{u} \quad \Leftrightarrow \quad \boldsymbol{\Sigma}_{xx}^{-1} \mathbf{x} = \mathbf{G}^T \boldsymbol{\Sigma}_{yy}^{-1} \mathbf{y}. \quad (2.66)$$

We then take expectations:

$$\boldsymbol{\Sigma}_{xx}^{-1} \boldsymbol{\mu}_x = E[\mathbf{v}] = E[\mathbf{G}^T \mathbf{u}] = \mathbf{G}^T E[\mathbf{u}] = \mathbf{G}^T \boldsymbol{\Sigma}_{yy}^{-1} \boldsymbol{\mu}_y, \quad (2.67a)$$

$$\begin{aligned} \boldsymbol{\Sigma}_{xx}^{-1} &= E[(\mathbf{v} - \boldsymbol{\Sigma}_{xx}^{-1} \boldsymbol{\mu}_x)(\mathbf{v} - \boldsymbol{\Sigma}_{xx}^{-1} \boldsymbol{\mu}_x)^T] \\ &= \mathbf{G}^T E[(\mathbf{u} - \boldsymbol{\Sigma}_{yy}^{-1} \boldsymbol{\mu}_y)(\mathbf{u} - \boldsymbol{\Sigma}_{yy}^{-1} \boldsymbol{\mu}_y)^T] \mathbf{G} = \mathbf{G}^T \boldsymbol{\Sigma}_{yy}^{-1} \mathbf{G}. \end{aligned} \quad (2.67b)$$

Note that if $\boldsymbol{\Sigma}_{xx}^{-1}$ is not full rank, we cannot actually recover $\boldsymbol{\Sigma}_{xx}$ and $\boldsymbol{\mu}_x$ and must keep them in information form. However, multiple such estimates can be fused together, which is the subject of the next section.

2.2.6 Normalized Product of Gaussians

We now discuss a useful property of Gaussian PDFs; the normalized product (see Section 2.1.6) of K Gaussian PDFs is also a Gaussian PDF:

$$\begin{aligned} &\exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})\right) \\ &\equiv \eta \prod_{k=1}^K \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)\right), \end{aligned} \quad (2.68)$$

where

$$\boldsymbol{\Sigma}^{-1} = \sum_{k=1}^K \boldsymbol{\Sigma}_k^{-1}, \quad (2.69a)$$

$$\boldsymbol{\Sigma}^{-1} \boldsymbol{\mu} = \sum_{k=1}^K \boldsymbol{\Sigma}_k^{-1} \boldsymbol{\mu}_k, \quad (2.69b)$$

and η is a normalization constant to enforce the axiom of total probability. The normalized product of Gaussians comes up when fusing multiple estimates together. A one-dimensional example is provided in Figure 2.3.

We also have that

$$\begin{aligned} &\exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})\right) \\ &\equiv \eta \prod_{k=1}^K \exp\left(-\frac{1}{2}(\mathbf{G}_k \mathbf{x} - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1}(\mathbf{G}_k \mathbf{x} - \boldsymbol{\mu}_k)\right), \end{aligned} \quad (2.70)$$

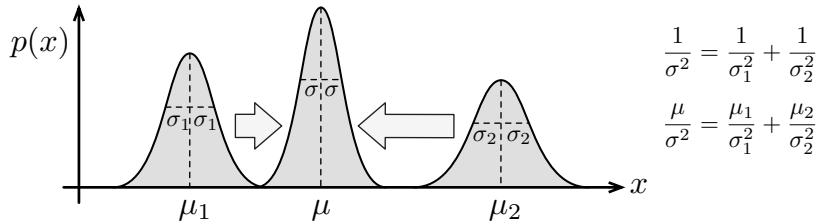


Figure 2.3 The normalized product of two one-dimensional Gaussian PDFs is another one-dimensional Gaussian PDF.

where

$$\Sigma^{-1} = \sum_{k=1}^K \mathbf{G}_k^T \Sigma_k^{-1} \mathbf{G}_k, \quad (2.71a)$$

$$\Sigma^{-1} \boldsymbol{\mu} = \sum_{k=1}^K \mathbf{G}_k^T \Sigma_k^{-1} \boldsymbol{\mu}_k, \quad (2.71b)$$

in the case that the matrices, $\mathbf{G}_k \in \mathbb{R}^{M_k \times N}$, are present, with $M_k \leq N$. Again, η is a normalization constant. We also note that this generalizes a result from the previous section.

2.2.7 Sherman-Morrison-Woodbury Identity

We will require the *Sherman-Morrison-Woodbury (SMW)* (Sherman and Morrison, 1949, 1950; Woodbury, 1950) matrix identity (sometimes called the *matrix inversion lemma*) in what follows. There are actually four different identities that come from a single derivation.

We start by noting that we can factor a matrix into either a *lower-diagonal-upper (LDU)* or *upper-diagonal-lower (UDL)* form, as follows:

$$\begin{aligned} & \begin{bmatrix} \mathbf{A}^{-1} & -\mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 \\ \mathbf{C}\mathbf{A} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{A}^{-1} & 0 \\ 0 & \mathbf{D} + \mathbf{C}\mathbf{A}\mathbf{B} \end{bmatrix} \begin{bmatrix} 1 & -\mathbf{A}\mathbf{B} \\ 0 & 1 \end{bmatrix} \quad (\text{LDU}) \\ &= \begin{bmatrix} 1 & -\mathbf{B}\mathbf{D}^{-1} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{A}^{-1} + \mathbf{B}\mathbf{D}^{-1}\mathbf{C} & 0 \\ 0 & \mathbf{D} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \mathbf{D}^{-1}\mathbf{C} & 1 \end{bmatrix}. \quad (\text{UDL}) \end{aligned} \quad (2.72)$$

We then invert each of these forms. For the LDU we have

$$\begin{aligned} & \begin{bmatrix} \mathbf{A}^{-1} & -\mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}^{-1} \\ &= \begin{bmatrix} 1 & \mathbf{A}\mathbf{B} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{A} & 0 \\ 0 & (\mathbf{D} + \mathbf{C}\mathbf{A}\mathbf{B})^{-1} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -\mathbf{C}\mathbf{A} & 1 \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{A} - \mathbf{A}\mathbf{B}(\mathbf{D} + \mathbf{C}\mathbf{A}\mathbf{B})^{-1}\mathbf{C}\mathbf{A} & \mathbf{A}\mathbf{B}(\mathbf{D} + \mathbf{C}\mathbf{A}\mathbf{B})^{-1} \\ -(\mathbf{D} + \mathbf{C}\mathbf{A}\mathbf{B})^{-1}\mathbf{C}\mathbf{A} & (\mathbf{D} + \mathbf{C}\mathbf{A}\mathbf{B})^{-1} \end{bmatrix}. \quad (2.73) \end{aligned}$$

The SMW formula is named for American statisticians Jack Sherman, Winifred J. Morrison, and Max A. Woodbury, but was independently presented by English mathematician W. J. Duncan, American statisticians L. Guttman and M. S. Bartlett, and possibly others.

For the UDL we have

$$\begin{aligned}
& \begin{bmatrix} \mathbf{A}^{-1} & -\mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}^{-1} \\
&= \begin{bmatrix} 1 & 0 \\ -\mathbf{D}^{-1}\mathbf{C} & 1 \end{bmatrix} \begin{bmatrix} (\mathbf{A}^{-1} + \mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1} & 0 \\ 0 & \mathbf{D}^{-1} \end{bmatrix} \begin{bmatrix} 1 & \mathbf{B}\mathbf{D}^{-1} \\ 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} (\mathbf{A}^{-1} + \mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1} & (\mathbf{A}^{-1} + \mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1}\mathbf{B}\mathbf{D}^{-1} \\ -\mathbf{D}^{-1}\mathbf{C}(\mathbf{A}^{-1} + \mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1} & \mathbf{D}^{-1} - \mathbf{D}^{-1}\mathbf{C}(\mathbf{A}^{-1} + \mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1}\mathbf{B}\mathbf{D}^{-1} \end{bmatrix}.
\end{aligned} \tag{2.74}$$

Comparing the blocks of (2.73) and (2.74), we have the following identities:

$$(\mathbf{A}^{-1} + \mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1} \equiv \mathbf{A} - \mathbf{A}\mathbf{B}(\mathbf{D} + \mathbf{C}\mathbf{A}\mathbf{B})^{-1}\mathbf{C}\mathbf{A}, \tag{2.75a}$$

$$(\mathbf{D} + \mathbf{C}\mathbf{A}\mathbf{B})^{-1} \equiv \mathbf{D}^{-1} - \mathbf{D}^{-1}\mathbf{C}(\mathbf{A}^{-1} + \mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1}\mathbf{B}\mathbf{D}^{-1}, \tag{2.75b}$$

$$\mathbf{A}\mathbf{B}(\mathbf{D} + \mathbf{C}\mathbf{A}\mathbf{B})^{-1} \equiv (\mathbf{A}^{-1} + \mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1}\mathbf{B}\mathbf{D}^{-1}, \tag{2.75c}$$

$$(\mathbf{D} + \mathbf{C}\mathbf{A}\mathbf{B})^{-1}\mathbf{C}\mathbf{A} \equiv \mathbf{D}^{-1}\mathbf{C}(\mathbf{A}^{-1} + \mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1}. \tag{2.75d}$$

These are all used frequently when manipulating expressions involving the covariance matrices associated with Gaussian PDFs.

2.2.8 Passing a Gaussian through a Nonlinearity

We now examine the process of passing a Gaussian PDF through a stochastic nonlinearity, namely, computing

$$p(\mathbf{y}) = \int_{-\infty}^{\infty} p(\mathbf{y}|\mathbf{x})p(\mathbf{x})d\mathbf{x}, \tag{2.76}$$

where we have that

$$p(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{g}(\mathbf{x}), \mathbf{R}), \tag{2.77a}$$

$$p(\mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}_x, \boldsymbol{\Sigma}_{xx}), \tag{2.77b}$$

and $\mathbf{g}(\cdot)$ is a nonlinear map, $\mathbf{g} : \mathbf{x} \mapsto \mathbf{y}$, that is then corrupted by zero-mean Gaussian noise with covariance, \mathbf{R} . We will require this type of stochastic nonlinearity when modelling sensors later on. Passing a Gaussian through this type of function is required, for example, in the denominator when carrying out full Bayesian inference.

Scalar Deterministic Case via Change of Variables

Let us first look at a simplified version where x is scalar and the nonlinear function, $g(\cdot)$, is deterministic (i.e., $R = 0$). We begin with a Gaussian random variable, $x \in \mathbb{R}^1$:

$$x \sim \mathcal{N}(0, \sigma^2). \tag{2.78}$$

For the PDF on x we have

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\frac{x^2}{\sigma^2}\right). \quad (2.79)$$

Now consider the nonlinear mapping,

$$y = \exp(x), \quad (2.80)$$

which is invertible:

$$x = \ln(y). \quad (2.81)$$

The infinitesimal integration volumes for x and y are then related by

$$dy = \exp(x) dx, \quad (2.82)$$

or

$$dx = \frac{1}{y} dy. \quad (2.83)$$

According to the axiom of total probability, we have

$$\begin{aligned} 1 &= \int_{-\infty}^{\infty} p(x) dx \\ &= \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\frac{x^2}{\sigma^2}\right) dx \\ &= \int_0^{\infty} \underbrace{\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\frac{(\ln(y))^2}{\sigma^2}\right)}_{p(y)} \frac{1}{y} dy \\ &= \int_0^{\infty} p(y) dy, \end{aligned} \quad (2.84)$$

giving us the exact expression for $p(y)$, which is plotted in Figure 2.4 for $\sigma^2 = 1$ as the black curve; the area under this curve, from $y = 0$ to ∞ is 1. The gray histogram is a numerical approximation of the PDF

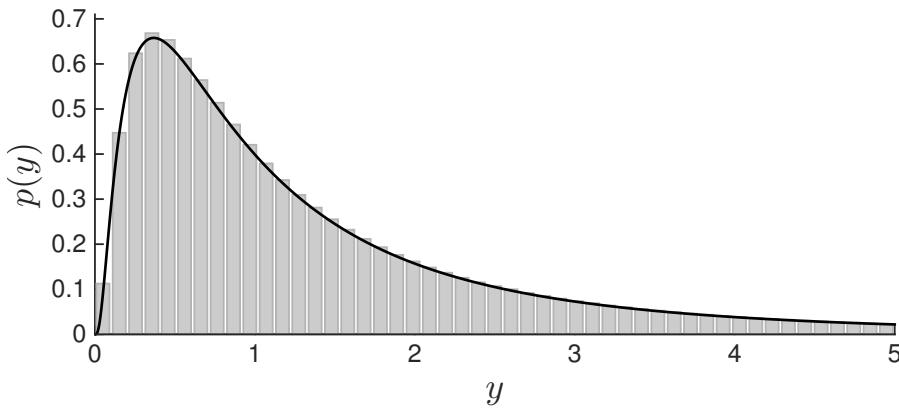
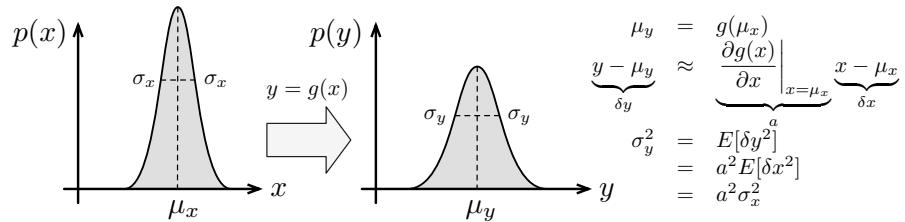


Figure 2.4 The PDF, $p(y)$, resulting from passing $p(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}x^2\right)$ through the nonlinearity, $y = \exp(x)$.

Figure 2.5
Passing a one-dimensional Gaussian through a deterministic nonlinear function, $g(\cdot)$. Here we linearize the nonlinearity in order to propagate the variance approximately.



generated by sampling x a large number of times and passing these through the nonlinearity individually, then binning. These approaches agree very well, validating our method of changing variables.

Note that $p(y)$ is no longer Gaussian owing to the nonlinear change of variables. We can verify numerically that the area under this function is indeed 1 (i.e., it is a valid PDF). It is worth noting that had we not been careful about handling the change of variables and including the $\frac{1}{y}$ factor, we would not have a valid PDF.

General Case via Linearization

Unfortunately, (2.76) cannot be computed in closed form for every $\mathbf{g}(\cdot)$ and becomes more difficult in the multivariate case than the scalar one. Moreover, when the nonlinearity is stochastic (i.e., $\mathbf{R} > 0$), our mapping will never be invertible due to the extra input coming from the noise, so we need a different way to transform our Gaussian. There are several different ways to do this, and in this section, we look at the most common one, *linearization*.

We linearize the nonlinear map such that

$$\begin{aligned}\mathbf{g}(\mathbf{x}) &\approx \boldsymbol{\mu}_y + \mathbf{G}(\mathbf{x} - \boldsymbol{\mu}_x), \\ \mathbf{G} &= \left. \frac{\partial \mathbf{g}(\mathbf{x})}{\partial \mathbf{x}} \right|_{\mathbf{x}=\boldsymbol{\mu}_x}, \\ \boldsymbol{\mu}_y &= \mathbf{g}(\boldsymbol{\mu}_x),\end{aligned}\tag{2.85}$$

where \mathbf{G} is the Jacobian of $\mathbf{g}(\cdot)$, with respect to \mathbf{x} . This allows us to then pass the Gaussian through the linearized function in closed form; it is an approximation that works well for mildly nonlinear maps.

Figure 2.5 depicts the process of passing a one-dimensional Gaussian PDF through a deterministic nonlinear function, $g(\cdot)$, that has been linearized. In general, we will be making an inference though a stochastic function, one that introduces additional noise.

Returning to (2.76), we have that

$$\begin{aligned}
p(\mathbf{y}) &= \int_{-\infty}^{\infty} p(\mathbf{y}|\mathbf{x})p(\mathbf{x})d\mathbf{x} \\
&= \eta \int_{-\infty}^{\infty} \exp \left(-\frac{1}{2} (\mathbf{y} - (\boldsymbol{\mu}_y + \mathbf{G}(\mathbf{x} - \boldsymbol{\mu}_x)))^T \right. \\
&\quad \times \mathbf{R}^{-1} (\mathbf{y} - (\boldsymbol{\mu}_y + \mathbf{G}(\mathbf{x} - \boldsymbol{\mu}_x))) \Big) \\
&\quad \times \exp \left(-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_x)^T \boldsymbol{\Sigma}_{xx}^{-1} (\mathbf{x} - \boldsymbol{\mu}_x) \right) d\mathbf{x} \\
&= \eta \exp \left(-\frac{1}{2} (\mathbf{y} - \boldsymbol{\mu}_y)^T \mathbf{R}^{-1} (\mathbf{y} - \boldsymbol{\mu}_y) \right) \\
&\quad \times \int_{-\infty}^{\infty} \exp \left(-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_x)^T (\boldsymbol{\Sigma}_{xx}^{-1} + \mathbf{G}^T \mathbf{R}^{-1} \mathbf{G}) (\mathbf{x} - \boldsymbol{\mu}_x) \right) \\
&\quad \times \exp ((\mathbf{y} - \boldsymbol{\mu}_y)^T \mathbf{R}^{-1} \mathbf{G}(\mathbf{x} - \boldsymbol{\mu}_x)) d\mathbf{x},
\end{aligned} \tag{2.86}$$

where η is a normalization constant. Defining \mathbf{F} such that

$$\mathbf{F}^T (\mathbf{G}^T \mathbf{R}^{-1} \mathbf{G} + \boldsymbol{\Sigma}_{xx}^{-1}) = \mathbf{R}^{-1} \mathbf{G}, \tag{2.87}$$

we may complete the square for the part inside the integral such that

$$\begin{aligned}
&\exp \left(-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_x)^T (\boldsymbol{\Sigma}_{xx}^{-1} + \mathbf{G}^T \mathbf{R}^{-1} \mathbf{G}) (\mathbf{x} - \boldsymbol{\mu}_x) \right) \\
&\quad \times \exp ((\mathbf{y} - \boldsymbol{\mu}_y)^T \mathbf{R}^{-1} \mathbf{G}(\mathbf{x} - \boldsymbol{\mu}_x)) \\
&= \exp \left(-\frac{1}{2} ((\mathbf{x} - \boldsymbol{\mu}_x) - \mathbf{F}(\mathbf{y} - \boldsymbol{\mu}_y))^T \right. \\
&\quad \times (\mathbf{G}^T \mathbf{R}^{-1} \mathbf{G} + \boldsymbol{\Sigma}_{xx}^{-1}) ((\mathbf{x} - \boldsymbol{\mu}_x) - \mathbf{F}(\mathbf{y} - \boldsymbol{\mu}_y)) \Big) \\
&\quad \times \exp \left(\frac{1}{2} (\mathbf{y} - \boldsymbol{\mu}_y)^T \mathbf{F}^T (\mathbf{G}^T \mathbf{R}^{-1} \mathbf{G} + \boldsymbol{\Sigma}_{xx}^{-1}) \mathbf{F}(\mathbf{y} - \boldsymbol{\mu}_y) \right).
\end{aligned} \tag{2.88}$$

The second factor is independent of \mathbf{x} and may be brought outside of the integral. The remaining integral (the first factor) is exactly Gaussian in \mathbf{x} and thus will integrate (over \mathbf{x}) to a constant and thus can be

absorbed in the constant η . Thus, for $p(\mathbf{y})$, we have

$$\begin{aligned}
 p(\mathbf{y}) &= \rho \exp \left(-\frac{1}{2} (\mathbf{y} - \boldsymbol{\mu}_y)^T \right. \\
 &\quad \times \left. (\mathbf{R}^{-1} - \mathbf{F}^T (\mathbf{G}^T \mathbf{R}^{-1} \mathbf{G} + \boldsymbol{\Sigma}_{xx}^{-1}) \mathbf{F}) (\mathbf{y} - \boldsymbol{\mu}_y) \right) \\
 &= \rho \exp \left(-\frac{1}{2} (\mathbf{y} - \boldsymbol{\mu}_y)^T \right. \\
 &\quad \times \left. \underbrace{(\mathbf{R}^{-1} - \mathbf{R}^{-1} \mathbf{G} (\mathbf{G}^T \mathbf{R}^{-1} \mathbf{G} + \boldsymbol{\Sigma}_{xx}^{-1})^{-1} \mathbf{G}^T \mathbf{R}^{-1}) (\mathbf{y} - \boldsymbol{\mu}_y)}_{(\mathbf{R} + \mathbf{G} \boldsymbol{\Sigma}_{xx} \mathbf{G}^T)^{-1} \text{ by (2.75)}} \right) \\
 &= \rho \exp \left(-\frac{1}{2} (\mathbf{y} - \boldsymbol{\mu}_y)^T (\mathbf{R} + \mathbf{G} \boldsymbol{\Sigma}_{xx} \mathbf{G}^T)^{-1} (\mathbf{y} - \boldsymbol{\mu}_y) \right), \quad (2.89)
 \end{aligned}$$

where ρ is the new normalization constant. This is a Gaussian for \mathbf{y} :

$$\mathbf{y} \sim \mathcal{N}(\boldsymbol{\mu}_y, \boldsymbol{\Sigma}_{yy}) = \mathcal{N}(\mathbf{g}(\boldsymbol{\mu}_x), \mathbf{R} + \mathbf{G} \boldsymbol{\Sigma}_{xx} \mathbf{G}^T). \quad (2.90)$$

As we will see later, the two equations (2.70) and (2.89) constitute the observation and predictive steps of the classic discrete-time (extended) Kalman filter (Kalman, 1960b). These two steps may be thought of as the creation and destruction of information in the filter, respectively.

2.2.9 Shannon Information of a Gaussian

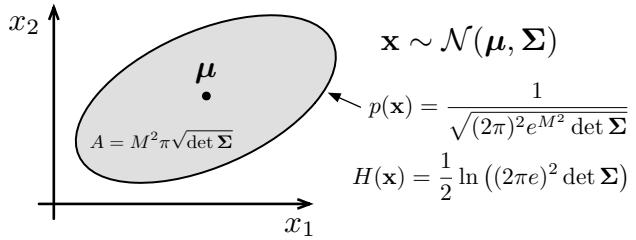
In the case of a Gaussian PDF, we have for the Shannon information:

$$\begin{aligned}
 H(\mathbf{x}) &= - \int_{-\infty}^{\infty} p(\mathbf{x}) \ln p(\mathbf{x}) d\mathbf{x} \\
 &= - \int_{-\infty}^{\infty} p(\mathbf{x}) \left(-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) - \ln \sqrt{(2\pi)^N \det \boldsymbol{\Sigma}} \right) d\mathbf{x} \\
 &= \frac{1}{2} \ln ((2\pi)^N \det \boldsymbol{\Sigma}) + \int_{-\infty}^{\infty} \frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) p(\mathbf{x}) d\mathbf{x} \\
 &= \frac{1}{2} \ln ((2\pi)^N \det \boldsymbol{\Sigma}) + \frac{1}{2} E [(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})], \quad (2.91)
 \end{aligned}$$

Prasanta Chandra Mahalanobis (1893-1972) was an Indian scientist and applied statistician known for this measure of statistical distance (Mahalanobis, 1936).

where we have written the second term using an expectation. In fact, this term is exactly a squared *Mahalanobis distance*, which is like a squared Euclidean distance but weighted in the middle by the inverse covariance matrix. A nice property of this quadratic function inside the expectation allows us to rewrite it using the (linear) *trace* operator from linear algebra:

$$(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) = \text{tr} (\boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) (\mathbf{x} - \boldsymbol{\mu})^T). \quad (2.92)$$



Since the expectation is also a linear operator, we may interchange the order of the expectation and trace arriving at

$$\begin{aligned}
 E [(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})] &= \text{tr} (E [\boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^T]) \\
 &= \text{tr} \left(\boldsymbol{\Sigma}^{-1} \underbrace{E [(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^T]}_{\boldsymbol{\Sigma}} \right) \\
 &= \text{tr} (\boldsymbol{\Sigma}^{-1} \boldsymbol{\Sigma}) \\
 &= \text{tr } \mathbf{1} \\
 &= N,
 \end{aligned} \tag{2.93}$$

which is just the dimension of the variable. Substituting this back into our expression for Shannon information, we have

$$\begin{aligned}
 H(\mathbf{x}) &= \frac{1}{2} \ln ((2\pi)^N \det \boldsymbol{\Sigma}) + \frac{1}{2} E [(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})] \\
 &= \frac{1}{2} \ln ((2\pi)^N \det \boldsymbol{\Sigma}) + \frac{1}{2} N \\
 &= \frac{1}{2} (\ln ((2\pi)^N \det \boldsymbol{\Sigma}) + N \ln e) \\
 &= \frac{1}{2} \ln ((2\pi e)^N \det \boldsymbol{\Sigma}),
 \end{aligned} \tag{2.94}$$

which is purely a function of $\boldsymbol{\Sigma}$, the covariance matrix of the Gaussian PDF. In fact, geometrically, we may interpret $\sqrt{\det \boldsymbol{\Sigma}}$ as proportional to the volume of the *uncertainty ellipsoid* formed by the Gaussian. Figure 2.6 shows the uncertainty ellipse for a two-dimensional Gaussian.

Note that along the boundary of the uncertainty ellipse, $p(\mathbf{x})$ is constant. To see this, consider that the points along this ellipse must satisfy

$$(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) = M^2, \tag{2.95}$$

where M is a factor applied to scale the nominal covariance so we have the $M = 1, 2, 3, \dots$ equiprobable contours. In this case we have that

$$p(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^N e^{M^2} \det \boldsymbol{\Sigma}}}, \tag{2.96}$$

on the M th ellipsoid surface.

Figure 2.6
Uncertainty ellipse for a two-dimensional Gaussian PDF. The geometric area inside the ellipse is $A = M^2 \pi \sqrt{\det \boldsymbol{\Sigma}}$. The Shannon information expression is provided for comparison.

2.2.10 Mutual Information of a Joint Gaussian PDF

Assume we have a joint Gaussian for variables $\mathbf{x} \in \mathbb{R}^N$ and $\mathbf{y} \in \mathbb{R}^M$ given by

$$p(\mathbf{x}, \mathbf{y}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_x \\ \boldsymbol{\mu}_y \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{xx} & \boldsymbol{\Sigma}_{xy} \\ \boldsymbol{\Sigma}_{yx} & \boldsymbol{\Sigma}_{yy} \end{bmatrix}\right). \quad (2.97)$$

By inserting (2.94) into (2.29) we can easily see that the mutual information for the joint Gaussian is given by

$$\begin{aligned} I(\mathbf{x}, \mathbf{y}) &= \frac{1}{2} \ln((2\pi e)^N \det \boldsymbol{\Sigma}_{xx}) + \frac{1}{2} \ln((2\pi e)^M \det \boldsymbol{\Sigma}_{yy}) \\ &\quad - \frac{1}{2} \ln((2\pi e)^{M+N} \det \boldsymbol{\Sigma}) \\ &= -\frac{1}{2} \ln\left(\frac{\det \boldsymbol{\Sigma}}{\det \boldsymbol{\Sigma}_{xx} \det \boldsymbol{\Sigma}_{yy}}\right). \end{aligned} \quad (2.98)$$

James Joseph Sylvester (1814-1897) was an English mathematician who made fundamental contributions to matrix theory, invariant theory, number theory, partition theory, and combinatorics. This theorem says that $\det(\mathbf{1} - \mathbf{AB}) = \det(\mathbf{1} - \mathbf{BA})$, even when \mathbf{A} and \mathbf{B} are not square.

Looking back to (2.50), we can also note that

$$\begin{aligned} \det \boldsymbol{\Sigma} &= \det \boldsymbol{\Sigma}_{xx} \det (\boldsymbol{\Sigma}_{yy} - \boldsymbol{\Sigma}_{yx} \boldsymbol{\Sigma}_{xx}^{-1} \boldsymbol{\Sigma}_{xy}) \\ &= \det \boldsymbol{\Sigma}_{yy} \det (\boldsymbol{\Sigma}_{xx} - \boldsymbol{\Sigma}_{xy} \boldsymbol{\Sigma}_{yy}^{-1} \boldsymbol{\Sigma}_{yx}). \end{aligned} \quad (2.99)$$

Inserting this into the above, we have

$$\begin{aligned} I(\mathbf{x}, \mathbf{y}) &= -\frac{1}{2} \ln \det(\mathbf{1} - \boldsymbol{\Sigma}_{xx}^{-1} \boldsymbol{\Sigma}_{xy} \boldsymbol{\Sigma}_{yy}^{-1} \boldsymbol{\Sigma}_{yx}) \\ &= -\frac{1}{2} \ln \det(\mathbf{1} - \boldsymbol{\Sigma}_{yy}^{-1} \boldsymbol{\Sigma}_{yx} \boldsymbol{\Sigma}_{xx}^{-1} \boldsymbol{\Sigma}_{xy}), \end{aligned} \quad (2.100)$$

where the two versions can be seen to be equivalent through *Sylvester's determinant theorem*.

2.2.11 Cramér-Rao Lower Bound Applied to Gaussian PDFs

Suppose that we have K samples (i.e., measurements), $\mathbf{x}_{\text{meas},k} \in \mathbb{R}^N$, drawn from a Gaussian PDF. The K statistically independent random variables associated with these measurements are thus

$$(\forall k) \quad \mathbf{x}_k \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}). \quad (2.101)$$

The term *statistically independent* implies that $E[(\mathbf{x}_k - \boldsymbol{\mu})(\mathbf{x}_\ell - \boldsymbol{\mu})^T] = \mathbf{0}$ for $k \neq \ell$. Now suppose our goal is to estimate the mean of this PDF, $\boldsymbol{\mu}$, from the measurements, $\mathbf{x}_{\text{meas},1}, \dots, \mathbf{x}_{\text{meas},K}$. For the joint density of all the random variables, $\mathbf{x} = (\mathbf{x}_1, \dots, \mathbf{x}_K)$, we in fact have

$$\ln p(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = -\frac{1}{2}(\mathbf{x} - \mathbf{A}\boldsymbol{\mu})^T \mathbf{B}^{-1}(\mathbf{x} - \mathbf{A}\boldsymbol{\mu}) - \ln \sqrt{(2\pi)^{NK} \det \mathbf{B}}, \quad (2.102)$$

where

$$\mathbf{A} = \underbrace{\begin{bmatrix} \mathbf{1} & \mathbf{1} & \cdots & \mathbf{1} \end{bmatrix}^T}_{K \text{ blocks}}, \quad \mathbf{B} = \text{diag}(\underbrace{\boldsymbol{\Sigma}, \boldsymbol{\Sigma}, \dots, \boldsymbol{\Sigma}}_{K \text{ blocks}}). \quad (2.103)$$

In this case, we have

$$\frac{\partial \ln p(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})}{\partial \boldsymbol{\mu}}^T = \mathbf{A}^T \mathbf{B}^{-1} (\mathbf{x} - \mathbf{A}\boldsymbol{\mu}), \quad (2.104)$$

and thus the Fisher information matrix is

$$\begin{aligned} \mathbf{I}(\mathbf{x}|\boldsymbol{\mu}) &= E \left[\left(\frac{\partial \ln p(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})}{\partial \boldsymbol{\mu}} \right)^T \left(\frac{\partial \ln p(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})}{\partial \boldsymbol{\mu}} \right) \right] \\ &= E [\mathbf{A}^T \mathbf{B}^{-1} (\mathbf{x} - \mathbf{A}\boldsymbol{\mu}) (\mathbf{x} - \mathbf{A}\boldsymbol{\mu})^T \mathbf{B}^{-1} \mathbf{A}] \\ &= \mathbf{A}^T \mathbf{B}^{-1} \underbrace{E [(\mathbf{x} - \mathbf{A}\boldsymbol{\mu})(\mathbf{x} - \mathbf{A}\boldsymbol{\mu})^T]}_{\mathbf{B}} \mathbf{B}^{-1} \mathbf{A} \\ &= \mathbf{A}^T \mathbf{B}^{-1} \mathbf{A} \\ &= K \boldsymbol{\Sigma}^{-1}, \end{aligned} \quad (2.105)$$

which we can see is just K times the inverse covariance of the Gaussian density. The CRLB thus says

$$\text{cov}(\hat{\mathbf{x}} | \mathbf{x}_{\text{meas},1}, \dots, \mathbf{x}_{\text{meas},K}) \geq \frac{1}{K} \boldsymbol{\Sigma}. \quad (2.106)$$

In other words, the lower limit of the uncertainty in the estimate of the mean, $\hat{\mathbf{x}}$, becomes smaller and smaller the more measurements we have (as we would expect).

Note that in computing the CRLB, we did not need actually to specify the form of the unbiased estimator at all; the CRLB is the lower bound for any unbiased estimator. In this case, it is not hard to find an estimator that performs right at the CRLB:

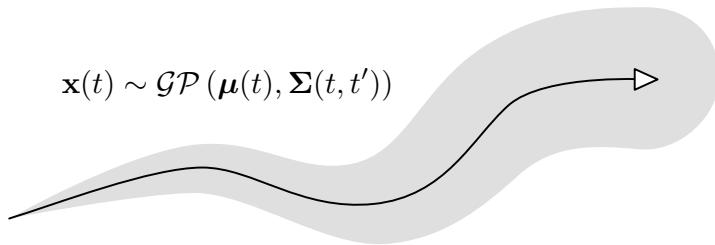
$$\hat{\mathbf{x}} = \frac{1}{K} \sum_{k=1}^K \mathbf{x}_{\text{meas},k}. \quad (2.107)$$

For the mean of this estimator we have

$$E[\hat{\mathbf{x}}] = E \left[\frac{1}{K} \sum_{k=1}^K \mathbf{x}_k \right] = \frac{1}{K} \sum_{k=1}^K E[\mathbf{x}_k] = \frac{1}{K} \sum_{k=1}^K \boldsymbol{\mu} = \boldsymbol{\mu}, \quad (2.108)$$

which shows that the estimator is indeed unbiased. For the covariance

Figure 2.7
 Continuous-time trajectories can be represented using Gaussian processes, which have a mean function (dark line) and a covariance function (shaded area).



we have

$$\begin{aligned}
 \text{cov}(\hat{\mathbf{x}} | \mathbf{x}_{\text{meas},1}, \dots, \mathbf{x}_{\text{meas},K}) &= E[(\hat{\mathbf{x}} - \boldsymbol{\mu})(\hat{\mathbf{x}} - \boldsymbol{\mu})^T] \\
 &= E\left[\left(\frac{1}{K} \sum_{k=1}^K \mathbf{x}_k - \boldsymbol{\mu}\right)\left(\frac{1}{K} \sum_{k=1}^K \mathbf{x}_k - \boldsymbol{\mu}\right)^T\right] \\
 &= \frac{1}{K^2} \sum_{k=1}^K \sum_{\ell=1}^K \underbrace{E[(\mathbf{x}_k - \boldsymbol{\mu})(\mathbf{x}_\ell - \boldsymbol{\mu})^T]}_{\boldsymbol{\Sigma} \text{ when } k = \ell, \mathbf{0} \text{ otherwise}} \\
 &= \frac{1}{K} \boldsymbol{\Sigma},
 \end{aligned} \tag{2.109}$$

which is right at the CRLB.

2.3 Gaussian Processes

We have already discussed Gaussian random variables and their associated PDFs. We write

$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}), \tag{2.110}$$

to say $\mathbf{x} \in \mathbb{R}^N$ is Gaussian. We will use this type of random variable extensively to represent discrete-time quantities. We will also want to talk about state quantities that are continuous functions of time, t . To do so, we need to introduce *Gaussian processes (GPs)* (Rasmussen and Williams, 2006). Figure 2.7 depicts a trajectory represented by a Gaussian process. There is a *mean function*, $\boldsymbol{\mu}(t)$, and a *covariance function*, $\boldsymbol{\Sigma}(t, t')$.

The idea is that the entire trajectory is a single random variable belonging to a class of functions. The closer a function is to the mean function, the more likely it is. The covariance function controls how smooth the function is by describing the correlation between two times, t and t' . We write

$$\mathbf{x}(t) \sim \mathcal{GP}(\boldsymbol{\mu}(t), \boldsymbol{\Sigma}(t, t')) \tag{2.111}$$

to indicate that a continuous-time trajectory is a *Gaussian process (GP)*. The GP concept generalizes beyond one-dimensional functions of time, but we will only have need of this special case.

If we want to consider a variable at a single particular time of interest, τ , then we can write

$$\mathbf{x}(\tau) \sim \mathcal{N}(\boldsymbol{\mu}(\tau), \boldsymbol{\Sigma}(\tau, \tau)), \quad (2.112)$$

where $\boldsymbol{\Sigma}(\tau, \tau)$ is now simply a covariance matrix. We have essentially marginalized out all of the other instants of time, leaving $\mathbf{x}(\tau)$ as a usual Gaussian random variable.

In general, a GP can take on many different forms. One particular GP that we will use frequently is the *zero-mean, white noise process*. For $\mathbf{w}(t)$ to be zero-mean, white noise, we write

$$\mathbf{w}(t) \sim \mathcal{GP}(\mathbf{0}, \mathbf{Q}\delta(t - t')), \quad (2.113)$$

where \mathbf{Q} is a *power spectral density matrix* and $\delta(\cdot)$ is *Dirac's delta function*. This is a *stationary* noise process since it depends only on the difference, $t - t'$.

We will return to GPs when we want to talk about state estimation in continuous time. We will show that estimation in this context can be viewed as an application of *Gaussian process regression* (Rasmussen and Williams, 2006).

Paul Adrien Maurice Dirac (1902-1984) was an English theoretical physicist who made fundamental contributions to the early development of both quantum mechanics and quantum electrodynamics.

2.4 Summary

The main take-away points from this chapter are as follows:

1. We will be using *probability density functions (PDFs)* over some continuous state space to represent how certain we are that a robot is in each possible state.
2. We often restrict ourselves to Gaussian PDFs to make the calculations easier.
3. We will frequently employ Bayes' rule to carry out so-called Bayesian inference as an approach to state estimation; we begin with a set of possible states (the prior) and narrow the possibilities based on actual measurements (the posterior).

The next chapter will introduce some of the classic linear-Gaussian, state estimation methods.

2.5 Exercises

- 2.5.1 Show for any two columns of the same length, \mathbf{u} and \mathbf{v} , that

$$\mathbf{u}^T \mathbf{v} = \text{tr}(\mathbf{v} \mathbf{u}^T).$$

- 2.5.2 Show that if two random variables, \mathbf{x} and \mathbf{y} , are statistically independent, then the Shannon information of the joint density,

$p(\mathbf{x}, \mathbf{y})$, is the sum of the Shannon informations of the individual densities, $p(\mathbf{x})$ and $p(\mathbf{y})$:

$$H(\mathbf{x}, \mathbf{y}) = H(\mathbf{x}) + H(\mathbf{y}).$$

2.5.3 For a Gaussian random variable, $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, show that

$$E[\mathbf{x}\mathbf{x}^T] = \boldsymbol{\Sigma} + \boldsymbol{\mu}\boldsymbol{\mu}^T.$$

2.5.4 For a Gaussian random variable, $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, show directly that

$$\boldsymbol{\mu} = E[\mathbf{x}] = \int_{-\infty}^{\infty} \mathbf{x} p(\mathbf{x}) d\mathbf{x}.$$

2.5.5 For a Gaussian random variable, $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, show directly that

$$\boldsymbol{\Sigma} = E[(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^T] = \int_{-\infty}^{\infty} (\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^T p(\mathbf{x}) d\mathbf{x}.$$

2.5.6 Show that the direct product of K statistically independent Gaussian PDFs, $\mathbf{x}_k \sim \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$, is also a Gaussian PDF:

$$\begin{aligned} & \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})\right) \\ & \equiv \eta \prod_{k=1}^K \exp\left(-\frac{1}{2}(\mathbf{x}_k - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1}(\mathbf{x}_k - \boldsymbol{\mu}_k)\right), \end{aligned}$$

where

$$\boldsymbol{\Sigma}^{-1} = \sum_{k=1}^K \boldsymbol{\Sigma}_k^{-1}, \quad \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu} = \sum_{k=1}^K \boldsymbol{\Sigma}_k^{-1} \boldsymbol{\mu}_k,$$

and η is a normalization constant to enforce the axiom of total probability.

2.5.7 Show that the weighted sum of K statistically independent random variables, \mathbf{x}_k , given by

$$\mathbf{x} = \sum_{k=1}^K w_k \mathbf{x}_k,$$

with $\sum_{k=1}^K w_k = 1$ and $w_k \geq 0$, has a PDF that satisfies the axiom of total probability and whose mean is given by

$$\boldsymbol{\mu} = \sum_{k=1}^K w_k \boldsymbol{\mu}_k,$$

where $\boldsymbol{\mu}_k$ is the mean of \mathbf{x}_k . Determine an expression for the covariance. Note that the random variables are not assumed to be Gaussian.

2.5.8 The random variable

$$y = \mathbf{x}^T \mathbf{x}$$

is *chi-squared* (of order K) when $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is length K . Show that the mean and variance are given by K and $2K$, respectively.
Hint: use Isserlis' theorem.

3

Linear-Gaussian Estimation

This chapter will introduce some of the classic results from estimation theory for linear models and Gaussian random variables, including the *Kalman filter (KF)* (Kalman, 1960b). We will begin with a batch, discrete-time estimation problem that will provide important insights into the nonlinear extension of the work in subsequent chapters. From the batch approach, we will show how the recursive methods can be developed. Finally, we will circle back to the more general case of handling continuous-time motion models and connect these to the discrete-time results as well as to Gaussian process regression from the machine-learning world. Classic books that cover linear estimation include Bryson (1975), Maybeck (1994), and Stengel (1994).

3.1 Batch Discrete-Time Estimation

We will begin by setting up the problem that we want to solve and then discuss methods of solution.

3.1.1 Problem Setup

In much of this chapter, we will consider *discrete-time, linear, time-varying* equations. We define the following motion and observation models:

$$\text{motion model: } \mathbf{x}_k = \mathbf{A}_{k-1} \mathbf{x}_{k-1} + \mathbf{v}_k + \mathbf{w}_k, \quad k = 1 \dots K \quad (3.1a)$$

$$\text{observation model: } \mathbf{y}_k = \mathbf{C}_k \mathbf{x}_k + \mathbf{n}_k, \quad k = 0 \dots K \quad (3.1b)$$

where k is the discrete-time index and K its maximum. The variables in (3.1) have the following meanings:

$$\begin{aligned} \text{system state : } & \mathbf{x}_k \in \mathbb{R}^N \\ \text{initial state : } & \mathbf{x}_0 \in \mathbb{R}^N \sim \mathcal{N}(\check{\mathbf{x}}_0, \check{\mathbf{P}}_0) \\ \text{input : } & \mathbf{v}_k \in \mathbb{R}^N \\ \text{process noise : } & \mathbf{w}_k \in \mathbb{R}^N \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_k) \\ \text{measurement : } & \mathbf{y}_k \in \mathbb{R}^M \\ \text{measurement noise : } & \mathbf{n}_k \in \mathbb{R}^M \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k) \end{aligned}$$

These are all *random variables*, except \mathbf{v}_k , which is deterministic¹. The noise variables and initial state knowledge are all assumed to be uncorrelated with one another (and with themselves at different timesteps). The matrix $\mathbf{A}_k \in \mathbb{R}^{N \times N}$ is called the *transition matrix*. The matrix $\mathbf{C}_k \in \mathbb{R}^{M \times N}$ is called the *observation matrix*.

Although we want to know the state of the system (at all times), we only have access to the following quantities, and must base our *estimate*, $\hat{\mathbf{x}}_k$, on just this information:

- (i) The initial state knowledge, $\check{\mathbf{x}}_0$, and the associated covariance matrix, $\check{\mathbf{P}}_0$; sometimes we do not have this piece of information and must do without².
- (ii) The inputs, \mathbf{v}_k , which typically come from the output of our controller and so are known³; we also have the associated process noise covariance, \mathbf{Q}_k .
- (iii) The measurements, $\mathbf{y}_{k,\text{meas}}$, which are *realizations* of the associated random variables, \mathbf{y}_k , and the associated covariance matrix, \mathbf{R}_k .

Based on the models in the previous section, we define the state estimation problem as follows:

The problem of state estimation is to come up with an estimate, $\hat{\mathbf{x}}_k$, of the true state of a system, at one or more timesteps, k , given knowledge of the initial state, $\check{\mathbf{x}}_0$, a sequence of measurements, $\mathbf{y}_{0:K,\text{meas}}$, a sequence of inputs, $\mathbf{v}_{1:K}$, as well as knowledge of the system's motion and observation models.

The rest of this chapter will present a suite of techniques for addressing this state estimation problem. Our approach will always be not only to attempt to come up with a state estimate but also to quantify the uncertainty in that estimate.

To set ourselves up for what is to follow in the later chapters on nonlinear estimation, we will begin by formulating a batch *linear-Gaussian (LG)* estimation problem. The batch solution is very useful for computing state estimates after the fact because it uses all the measurements in the estimation of all the states at once (hence the usage of ‘batch’). However, a batch method cannot be used in real time since we cannot employ future measurements to estimate past states. For this we will need recursive state estimators, which will be covered later in this chapter.

¹ Sometimes the input is specialized to be of the form $\mathbf{v}_k = \mathbf{B}_k \mathbf{u}_k$, where $\mathbf{u}_k \in \mathbb{R}^U$ is now the input and $\mathbf{B}_k \in \mathbb{R}^{N \times U}$ is called the *control matrix*. We will use this form as needed in our development.

² We will use $(\hat{\cdot})$ to indicate *posterior* estimates (including measurements) and $(\check{\cdot})$ to indicate *prior* estimates (not including measurements).

³ In robotics, this input is sometimes replaced by an interoceptive measurement. This is a bit of a dangerous thing to do since it then conflates two sources of uncertainty: process noise and measurement noise. If this is done, we must be careful to inflate \mathbf{Q} appropriately to reflect the two uncertainties.

To show the relationship between various concepts, we will set up the batch LG estimation problem using two different paradigms:

- (i) *Bayesian inference*; here we update a prior density over states (based on the initial state knowledge, inputs, and motion model) with our measurements, to produce a posterior (Gaussian) density over states.
- (ii) *Maximum A Posteriori (MAP)*; here we employ optimization to find the most likely posterior state given the information we have (initial state knowledge, measurements, inputs).

While these approaches are somewhat different in nature, it turns out that we arrive at the exact same answer for the LG problem. This is because the full Bayesian posterior is exactly Gaussian. Therefore, the optimization approach will find the maximum (i.e., mode) of a Gaussian, and this is the same as the mean. It is important to pursue these two avenues because when we move to nonlinear, non-Gaussian systems in subsequent chapters, the mean and mode of the posterior will no longer be the same and the two methods will arrive at different answers. We will start with the MAP optimization method as it is a bit easier to explain.

3.1.2 Maximum A Posteriori

In batch estimation, our goal is to solve the following MAP problem:

$$\hat{\mathbf{x}} = \arg \max_{\mathbf{x}} p(\mathbf{x} | \mathbf{v}, \mathbf{y}), \quad (3.2)$$

which is to say that we want to find the best single estimate for the state of the system (at all timesteps), $\hat{\mathbf{x}}$, given the prior information, \mathbf{v} , and measurements, \mathbf{y} ⁴. Note that we have

$$\begin{aligned} \mathbf{x} &= \mathbf{x}_{0:K} = (\mathbf{x}_0, \dots, \mathbf{x}_K), & \mathbf{v} &= (\check{\mathbf{x}}_0, \mathbf{v}_{1:K}) = (\check{\mathbf{x}}_0, \mathbf{v}_1, \dots, \mathbf{v}_K), \\ \mathbf{y} &= \mathbf{y}_{0:K} = (\mathbf{y}_0, \dots, \mathbf{y}_K), \end{aligned}$$

where the timestep range may be dropped for convenience of notation (when the range is the largest possible for that variable)⁵. Note that we have included the initial state information with the inputs to the system; together these define our prior over the state. The measurements serve to improve this prior information.

⁴ We will be a bit loose on notation here by dropping ‘meas’ from \mathbf{y}_{meas} .

⁵ We will sometimes refer to *lifted form* when discussing variables and equations over the entire trajectory rather than a single timestep. It should be clear when quantities are in lifted form as they will not have a subscript for the timestep.

We begin by rewriting the MAP estimate using Bayes' rule:

$$\begin{aligned}\hat{\mathbf{x}} &= \arg \max_{\mathbf{x}} p(\mathbf{x}|\mathbf{v}, \mathbf{y}) = \arg \max_{\mathbf{x}} \frac{p(\mathbf{y}|\mathbf{x}, \mathbf{v})p(\mathbf{x}|\mathbf{v})}{p(\mathbf{y}|\mathbf{v})} \\ &= \arg \max_{\mathbf{x}} p(\mathbf{y}|\mathbf{x})p(\mathbf{x}|\mathbf{v}),\end{aligned}\quad (3.3)$$

where we drop the denominator because it does not depend on \mathbf{x} . We also drop \mathbf{v} in $p(\mathbf{y}|\mathbf{x}, \mathbf{v})$ since it does not affect \mathbf{y} in our system if \mathbf{x} is known (see observation model).

A vital assumption that we are making is that all of the noise variables, \mathbf{w}_k and \mathbf{n}_k for $k = 0 \dots K$, are uncorrelated. This allows us to use Bayes' rule to factor $p(\mathbf{y}|\mathbf{x})$ in the following way:

$$p(\mathbf{y}|\mathbf{x}) = \prod_{k=0}^K p(\mathbf{y}_k | \mathbf{x}_k). \quad (3.4)$$

Furthermore, Bayes' rule allows us to factor $p(\mathbf{x}|\mathbf{v})$ as

$$p(\mathbf{x}|\mathbf{v}) = p(\mathbf{x}_0 | \check{\mathbf{x}}_0) \prod_{k=1}^K p(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{v}_k). \quad (3.5)$$

In this linear system, the component (Gaussian) densities are given by

$$\begin{aligned}p(\mathbf{x}_0 | \check{\mathbf{x}}_0) &= \frac{1}{\sqrt{(2\pi)^N \det \check{\mathbf{P}}_0}} \\ &\times \exp \left(-\frac{1}{2} (\mathbf{x}_0 - \check{\mathbf{x}}_0)^T \check{\mathbf{P}}_0^{-1} (\mathbf{x}_0 - \check{\mathbf{x}}_0) \right),\end{aligned}\quad (3.6a)$$

$$\begin{aligned}p(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{v}_k) &= \frac{1}{\sqrt{(2\pi)^M \det \mathbf{Q}_k}} \exp \left(-\frac{1}{2} (\mathbf{x}_k - \mathbf{A}_{k-1} \mathbf{x}_{k-1} - \mathbf{v}_k)^T \right. \\ &\quad \left. \times \mathbf{Q}_k^{-1} (\mathbf{x}_k - \mathbf{A}_{k-1} \mathbf{x}_{k-1} - \mathbf{v}_k) \right),\end{aligned}\quad (3.6b)$$

$$\begin{aligned}p(\mathbf{y}_k | \mathbf{x}_k) &= \frac{1}{\sqrt{(2\pi)^L \det \mathbf{R}_k}} \exp \left(-\frac{1}{2} (\mathbf{y}_k - \mathbf{C}_k \mathbf{x}_k)^T \right. \\ &\quad \left. \times \mathbf{R}_k^{-1} (\mathbf{y}_k - \mathbf{C}_k \mathbf{x}_k) \right).\end{aligned}\quad (3.6c)$$

Note that we must have $\check{\mathbf{P}}_0$, \mathbf{Q}_k , and \mathbf{R}_k invertible; they are in fact positive-definite by assumption and therefore invertible. To make the

optimization easier, we take the logarithm of both sides⁶:

$$\ln(p(\mathbf{y}|\mathbf{x})p(\mathbf{x}|\mathbf{v})) = \ln p(\mathbf{x}_0 | \check{\mathbf{x}}_0) + \sum_{k=1}^K \ln p(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{v}_k) + \sum_{k=0}^K \ln p(\mathbf{y}_k | \mathbf{x}_k), \quad (3.7)$$

where

$$\begin{aligned} \ln p(\mathbf{x}_0 | \check{\mathbf{x}}_0) &= -\frac{1}{2} (\mathbf{x}_0 - \check{\mathbf{x}}_0)^T \check{\mathbf{P}}_0^{-1} (\mathbf{x}_0 - \check{\mathbf{x}}_0) \\ &\quad - \underbrace{\frac{1}{2} \ln ((2\pi)^N \det \check{\mathbf{P}}_0)}_{\text{independent of } \mathbf{x}}, \end{aligned} \quad (3.8a)$$

$$\begin{aligned} \ln p(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{v}_k) &= -\frac{1}{2} (\mathbf{x}_k - \mathbf{A}_{k-1}\mathbf{x}_{k-1} - \mathbf{v}_k)^T \\ &\quad \times \mathbf{Q}_k^{-1} (\mathbf{x}_k - \mathbf{A}_{k-1}\mathbf{x}_{k-1} - \mathbf{v}_k) \\ &\quad - \underbrace{\frac{1}{2} \ln ((2\pi)^N \det \mathbf{Q}_k)}_{\text{independent of } \mathbf{x}}, \end{aligned} \quad (3.8b)$$

$$\begin{aligned} \ln p(\mathbf{y}_k | \mathbf{x}_k) &= -\frac{1}{2} (\mathbf{y}_k - \mathbf{C}_k \mathbf{x}_k)^T \mathbf{R}_k^{-1} (\mathbf{y}_k - \mathbf{C}_k \mathbf{x}_k) \\ &\quad - \underbrace{\frac{1}{2} \ln ((2\pi)^M \det \mathbf{R}_k)}_{\text{independent of } \mathbf{x}}. \end{aligned} \quad (3.8c)$$

Noticing that there are terms in (3.8) that do not depend on \mathbf{x} , we define the following quantities:

$$J_{v,k}(\mathbf{x}) = \begin{cases} \frac{1}{2} (\mathbf{x}_0 - \check{\mathbf{x}}_0)^T \check{\mathbf{P}}_0^{-1} (\mathbf{x}_0 - \check{\mathbf{x}}_0), & k = 0 \\ \frac{1}{2} (\mathbf{x}_k - \mathbf{A}_{k-1}\mathbf{x}_{k-1} - \mathbf{v}_k)^T \times \mathbf{Q}_k^{-1} (\mathbf{x}_k - \mathbf{A}_{k-1}\mathbf{x}_{k-1} - \mathbf{v}_k), & k = 1 \dots K \end{cases}, \quad (3.9a)$$

$$J_{y,k}(\mathbf{x}) = \frac{1}{2} (\mathbf{y}_k - \mathbf{C}_k \mathbf{x}_k)^T \mathbf{R}_k^{-1} (\mathbf{y}_k - \mathbf{C}_k \mathbf{x}_k), \quad k = 0 \dots K, \quad (3.9b)$$

which are all squared *Mahalanobis distances*. We then define an overall *objective function*, $J(\mathbf{x})$, that we will seek to minimize with respect to the *design parameter*, \mathbf{x} :

$$J(\mathbf{x}) = \sum_{k=0}^K (J_{v,k}(\mathbf{x}) + J_{y,k}(\mathbf{x})). \quad (3.10)$$

We will work with $J(\mathbf{x})$ as is, but note that it is possible to add all kinds of additional terms to this expression that will influence the solution for the best estimate (e.g., constraints, penalty terms). From an

⁶ A logarithm is a monotonically increasing function and therefore will not affect our optimization problem.

optimization perspective, we seek to solve the following problem:

$$\hat{\mathbf{x}} = \arg \min_{\mathbf{x}} J(\mathbf{x}), \quad (3.11)$$

which will result in the same solution for the best estimate, $\hat{\mathbf{x}}$, as (3.2). In other words, we are still finding the best estimate in order to maximize the likelihood of the state given all the data we have. This is an *unconstrained optimization problem* in that we do not have to satisfy any constraints on the design variable, \mathbf{x} .

To further simplify our problem, we make use of the fact that equations (3.9) are quadratic in \mathbf{x} . To make this more clear, we stack all the known data into a lifted column, \mathbf{z} , and recall that \mathbf{x} is also a tall column consisting of all the states:

$$\mathbf{z} = \begin{bmatrix} \dot{\mathbf{x}}_0 \\ \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_K \\ \mathbf{y}_0 \\ \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_K \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} \mathbf{x}_0 \\ \vdots \\ \mathbf{x}_K \end{bmatrix}. \quad (3.12)$$

We then define the following block-matrix quantities:

$$\mathbf{H} = \begin{bmatrix} \mathbf{1} & & & \\ -\mathbf{A}_0 & \mathbf{1} & & \\ & \ddots & \ddots & \\ & & -\mathbf{A}_{K-1} & \mathbf{1} \\ \hline \mathbf{C}_0 & & & \\ & \mathbf{C}_1 & & \\ & & \ddots & \\ & & & \mathbf{C}_K \end{bmatrix}, \quad (3.13a)$$

$$\mathbf{W} = \begin{bmatrix} \check{\mathbf{P}}_0 & & & \\ & \mathbf{Q}_1 & & \\ & & \ddots & \\ & & & \mathbf{Q}_K \\ \hline & & & \\ & & & \mathbf{R}_0 \\ & & & & \mathbf{R}_1 \\ & & & & & \ddots \\ & & & & & & \mathbf{R}_K \end{bmatrix}, \quad (3.13b)$$

where only non-zero blocks are shown. The solid partition lines are used to show the boundaries between the parts of the matrices relevant

to the prior, \mathbf{v} , and the measurements, \mathbf{y} , in the lifted data vector, \mathbf{z} . Under these definitions, we find that

$$J(\mathbf{x}) = \frac{1}{2} (\mathbf{z} - \mathbf{Hx})^T \mathbf{W}^{-1} (\mathbf{z} - \mathbf{Hx}), \quad (3.14)$$

which is exactly quadratic in \mathbf{x} . We note that we also have

$$p(\mathbf{z}|\mathbf{x}) = \eta \exp\left(-\frac{1}{2} (\mathbf{z} - \mathbf{Hx})^T \mathbf{W}^{-1} (\mathbf{z} - \mathbf{Hx})\right), \quad (3.15)$$

where η is a normalization constant.

Since $J(\mathbf{x})$ is exactly a paraboloid, we can find its minimum in closed form. Simply set the partial derivative with respect to the design variable, \mathbf{x} , to zero:

$$\frac{\partial J(\mathbf{x})}{\partial \mathbf{x}^T} \Big|_{\hat{\mathbf{x}}} = -\mathbf{H}^T \mathbf{W}^{-1} (\mathbf{z} - \mathbf{H}\hat{\mathbf{x}}) = \mathbf{0}, \quad (3.16a)$$

$$\Rightarrow (\mathbf{H}^T \mathbf{W}^{-1} \mathbf{H}) \hat{\mathbf{x}} = \mathbf{H}^T \mathbf{W}^{-1} \mathbf{z}. \quad (3.16b)$$

The solution of (3.16b), $\hat{\mathbf{x}}$, is the classic *batch least-squares* solution and is equivalent to the *fixed-interval smoother*⁷ from classic estimation theory. The *batch least-squares* solution employs the *pseudoinverse*⁸. Computationally, to solve this linear system of equations, we would never actually invert $\mathbf{H}^T \mathbf{W}^{-1} \mathbf{H}$ (even if it were densely populated). As we will see later, we have a special block-tridiagonal structure to $\mathbf{H}^T \mathbf{W}^{-1} \mathbf{H}$, and hence a sparse-equation solver can be used to solve this system efficiently⁹.

One intuitive explanation of the batch linear-Gaussian problem is that it is like a mass-spring system, as shown in Figure 3.1. Each term in the objective function represents energy stored in one of the springs, which varies as the masses' positions are shifted. The optimal posterior solution corresponds to the minimum-energy state.

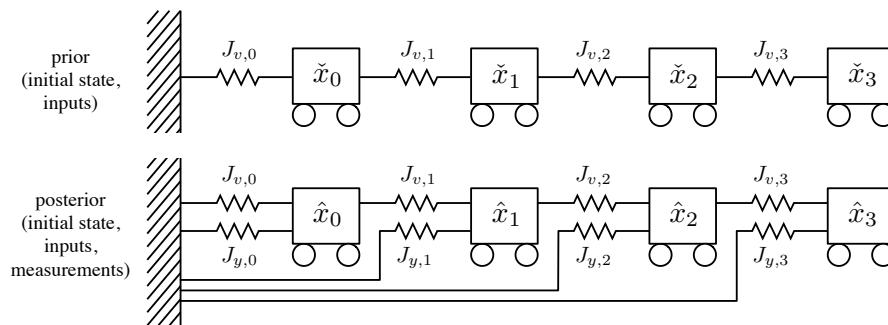


Figure 3.1 The batch linear-Gaussian problem is like a mass-spring system. Each term in the objective function represents energy stored in one of the springs, which varies as the carts' (i.e., masses) positions are shifted. The optimal posterior solution corresponds to the minimum energy state.

⁷ The fixed-interval smoother is usually presented in a recursive formulation. We will discuss this in more detail later.

⁸ Also called the *Moore-Penrose pseudoinverse*.

⁹ True for the problem posed; not true for all LG problems.

3.1.3 Bayesian Inference

Now that we have seen the optimization approach to batch LG estimation, we take a look at computing the full Bayesian posterior, $p(\mathbf{x}|\mathbf{v}, \mathbf{y})$, not just the maximum. This approach requires us to begin with a prior density over states, which we will then update based on the measurements.

In our case, a prior can be built up using the knowledge of the initial state, as well as the inputs to the system: $p(\mathbf{x}|\mathbf{v})$. We will use just the motion model to build this prior:

$$\mathbf{x}_k = \mathbf{A}_{k-1}\mathbf{x}_{k-1} + \mathbf{v}_k + \mathbf{w}_k. \quad (3.17)$$

In *lifted matrix form*¹⁰, we can write this as

$$\mathbf{x} = \mathbf{A}(\mathbf{v} + \mathbf{w}), \quad (3.18)$$

where \mathbf{w} is the lifted form of the initial state and process noise and

$$\mathbf{A} = \begin{bmatrix} \mathbf{1} & & & & \\ \mathbf{A}_0 & \mathbf{1} & & & \\ \mathbf{A}_1\mathbf{A}_0 & & \mathbf{A}_1 & & \mathbf{1} \\ \vdots & \vdots & \vdots & \ddots & \\ \mathbf{A}_{K-2} \cdots \mathbf{A}_0 & \mathbf{A}_{K-2} \cdots \mathbf{A}_1 & \mathbf{A}_{K-2} \cdots \mathbf{A}_2 & \cdots & \mathbf{1} \\ \mathbf{A}_{K-1} \cdots \mathbf{A}_0 & \mathbf{A}_{K-1} \cdots \mathbf{A}_1 & \mathbf{A}_{K-1} \cdots \mathbf{A}_2 & \cdots & \mathbf{A}_{K-1} \quad \mathbf{1} \end{bmatrix} \quad (3.19)$$

is the lifted transition matrix, which we see is lower-triangular. The lifted mean is then

$$\check{\mathbf{x}} = E[\mathbf{x}] = E[\mathbf{A}(\mathbf{v} + \mathbf{w})] = \mathbf{A}\mathbf{v}, \quad (3.20)$$

and lifted covariance is

$$\check{\mathbf{P}} = E[(\mathbf{x} - E[\mathbf{x}])(\mathbf{x} - E[\mathbf{x}])^T] = \mathbf{A}\mathbf{Q}\mathbf{A}^T, \quad (3.21)$$

where $\mathbf{Q} = E[\mathbf{w}\mathbf{w}^T] = \text{diag}(\check{\mathbf{P}}_0, \mathbf{Q}_1, \dots, \mathbf{Q}_K)$. Our prior can then be neatly expressed as

$$p(\mathbf{x}|\mathbf{v}) = \mathcal{N}(\check{\mathbf{x}}, \check{\mathbf{P}}) = \mathcal{N}(\mathbf{A}\mathbf{v}, \mathbf{A}\mathbf{Q}\mathbf{A}^T). \quad (3.22)$$

We next turn to the measurements.

The measurement model is

$$\mathbf{y}_k = \mathbf{C}_k\mathbf{x}_k + \mathbf{n}_k. \quad (3.23)$$

This can also be written in lifted form as

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{n}, \quad (3.24)$$

¹⁰ ‘Lifted’ here refers to the fact that we are considering what happens at the entire trajectory level.

where \mathbf{n} is the lifted form of the measurement noise and

$$\mathbf{C} = \text{diag}(\mathbf{C}_0, \mathbf{C}_1, \dots, \mathbf{C}_K) \quad (3.25)$$

is the lifted observation matrix.

The joint density of the prior lifted state and the measurements can now be written as

$$p(\mathbf{x}, \mathbf{y} | \mathbf{v}) = \mathcal{N}\left(\begin{bmatrix} \check{\mathbf{x}} \\ \mathbf{C}\check{\mathbf{x}} \end{bmatrix}, \begin{bmatrix} \check{\mathbf{P}} & \check{\mathbf{P}}\mathbf{C}^T \\ \mathbf{C}\check{\mathbf{P}} & \mathbf{C}\check{\mathbf{P}}\mathbf{C}^T + \mathbf{R} \end{bmatrix}\right), \quad (3.26)$$

where $\mathbf{R} = E[\mathbf{n}\mathbf{n}^T] = \text{diag}(\mathbf{R}_0, \mathbf{R}_1, \dots, \mathbf{R}_K)$. We can factor this according to

$$p(\mathbf{x}, \mathbf{y} | \mathbf{v}) = p(\mathbf{x} | \mathbf{v}, \mathbf{y})p(\mathbf{y} | \mathbf{v}). \quad (3.27)$$

We only care about the first factor, which is the full Bayesian posterior. This can be written, using the approach outlined in Section 2.2.3, as

$$p(\mathbf{x} | \mathbf{v}, \mathbf{y}) = \mathcal{N}\left(\check{\mathbf{x}} + \check{\mathbf{P}}\mathbf{C}^T(\mathbf{C}\check{\mathbf{P}}\mathbf{C}^T + \mathbf{R})^{-1}(\mathbf{y} - \mathbf{C}\check{\mathbf{x}}), \check{\mathbf{P}} - \check{\mathbf{P}}\mathbf{C}^T(\mathbf{C}\check{\mathbf{P}}\mathbf{C}^T + \mathbf{R})^{-1}\mathbf{C}\check{\mathbf{P}}\right). \quad (3.28)$$

Using the SMW identity from equations (2.75), this can be manipulated into the following form:

$$p(\mathbf{x} | \mathbf{v}, \mathbf{y}) = \mathcal{N}\left(\underbrace{(\check{\mathbf{P}}^{-1} + \mathbf{C}^T \mathbf{R}^{-1} \mathbf{C})^{-1} (\check{\mathbf{P}}^{-1} \check{\mathbf{x}} + \mathbf{C}^T \mathbf{R}^{-1} \mathbf{y})}_{\hat{\mathbf{x}}, \text{ mean}}, \underbrace{(\check{\mathbf{P}}^{-1} + \mathbf{C}^T \mathbf{R}^{-1} \mathbf{C})^{-1}}_{\hat{\mathbf{P}}, \text{ covariance}}\right). \quad (3.29)$$

We can actually implement a batch estimator based on this equation, since it represents the full Bayesian posterior, but this may not be efficient.

To see the connection to the optimization approach discussed earlier, we rearrange the mean expression to arrive at a linear system for $\hat{\mathbf{x}}$,

$$\underbrace{(\check{\mathbf{P}}^{-1} + \mathbf{C}^T \mathbf{R}^{-1} \mathbf{C})}_{\hat{\mathbf{P}}^{-1}} \hat{\mathbf{x}} = \check{\mathbf{P}}^{-1} \check{\mathbf{x}} + \mathbf{C}^T \mathbf{R}^{-1} \mathbf{y}, \quad (3.30)$$

and we see the inverse covariance appearing on the left-hand side. Substituting in $\check{\mathbf{x}} = \mathbf{A}\mathbf{v}$ and $\check{\mathbf{P}}^{-1} = (\mathbf{A}\mathbf{Q}\mathbf{A}^T)^{-1} = \mathbf{A}^{-T}\mathbf{Q}^{-1}\mathbf{A}^{-1}$ we can rewrite this as

$$\underbrace{(\mathbf{A}^{-T}\mathbf{Q}^{-1}\mathbf{A}^{-1} + \mathbf{C}^T \mathbf{R}^{-1} \mathbf{C})}_{\hat{\mathbf{P}}^{-1}} \hat{\mathbf{x}} = \mathbf{A}^{-T}\mathbf{Q}^{-1}\mathbf{v} + \mathbf{C}^T \mathbf{R}^{-1} \mathbf{y}. \quad (3.31)$$

We see that this requires computing \mathbf{A}^{-1} . It turns out this has a beautifully simple form¹¹,

$$\mathbf{A}^{-1} = \begin{bmatrix} \mathbf{1} & & & & \\ -\mathbf{A}_0 & \mathbf{1} & & & \\ & -\mathbf{A}_1 & \mathbf{1} & & \\ & & -\mathbf{A}_2 & \ddots & \\ & & & \ddots & \mathbf{1} \\ & & & & -\mathbf{A}_{K-1} & \mathbf{1} \end{bmatrix}, \quad (3.32)$$

which is still lower-triangular but also very sparse (only the main diagonal and the one below are non-zero). If we define

$$\mathbf{z} = \begin{bmatrix} \mathbf{v} \\ \mathbf{y} \end{bmatrix}, \quad \mathbf{H} = \begin{bmatrix} \mathbf{A}^{-1} \\ \mathbf{C} \end{bmatrix}, \quad \mathbf{W} = \begin{bmatrix} \mathbf{Q} & \\ & \mathbf{R} \end{bmatrix}, \quad (3.33)$$

we can rewrite our system of equations as

$$(\mathbf{H}^T \mathbf{W}^{-1} \mathbf{H}) \hat{\mathbf{x}} = \mathbf{H}^T \mathbf{W}^{-1} \mathbf{z}, \quad (3.34)$$

which is identical to the optimization solution discussed earlier.

Again, it must be stressed that the reason the Bayesian approach produces the same answer as the optimization solution for our LG estimation problem is that the full Bayesian posterior is exactly Gaussian and the mean and mode (i.e., maximum) of a Gaussian are one and the same.

3.1.4 Existence, Uniqueness, and Observability

Most of the classic LG estimation results can be viewed as a special case of (3.34). It is therefore important to ask when (3.34) has a unique solution, which is the subject of this section.

Examining (3.34), we have from basic linear algebra that $\hat{\mathbf{x}}$ will exist and be a unique solution if and only if $\mathbf{H}^T \mathbf{W}^{-1} \mathbf{H}$ is invertible, whereupon

$$\hat{\mathbf{x}} = (\mathbf{H}^T \mathbf{W}^{-1} \mathbf{H})^{-1} \mathbf{H}^T \mathbf{W}^{-1} \mathbf{z}. \quad (3.35)$$

The question is then, when is $\mathbf{H}^T \mathbf{W}^{-1} \mathbf{H}$ invertible? From linear algebra again, we know that a necessary and sufficient condition for invertibility is

$$\text{rank}(\mathbf{H}^T \mathbf{W}^{-1} \mathbf{H}) = N(K + 1), \quad (3.36)$$

because we have $\dim \mathbf{x} = N(K + 1)$. Since \mathbf{W}^{-1} is real symmetric

¹¹ The special sparsity of \mathbf{A}^{-1} is in fact critical to all classic LG results, as we will discuss later. This makes the left-hand side of (3.31) exactly block-tridiagonal. This means we can solve for $\hat{\mathbf{x}}$ in $O(K)$ time instead of the usual $O(K^3)$ time for solving linear systems. This leads to the popular recursive solution known as the Kalman filter/smooth. The sparsity comes from the fact that the system model obeys the Markov property.

positive-definite¹², we know that it can be dropped from the test so that we only need

$$\text{rank}(\mathbf{H}^T \mathbf{H}) = \text{rank}(\mathbf{H}^T) = N(K+1). \quad (3.37)$$

In other words, we need $N(K+1)$ linearly independent rows (or columns) in the matrix \mathbf{H}^T .

We now have two cases that should be considered:

- (i) We have good prior knowledge of the initial state, $\check{\mathbf{x}}_0$.
- (ii) We do not have good prior knowledge of the initial state.

The first case is much easier than the second.

Case (i): Knowledge of initial state

Writing out \mathbf{H}^T , our rank test takes the form

$$\begin{aligned} & \text{rank } \mathbf{H}^T \\ &= \text{rank} \left[\begin{array}{cccc|ccccc} \mathbf{1} & -\mathbf{A}_0^T & & & \mathbf{C}_0^T & & & & \\ & \mathbf{1} & -\mathbf{A}_1^T & & & \mathbf{C}_1^T & & & \\ & & \mathbf{1} & \ddots & & & \mathbf{C}_2^T & & \\ & & & \ddots & -\mathbf{A}_{K-1}^T & & & \ddots & \\ & & & & \mathbf{1} & & & & \mathbf{C}_K^T \end{array} \right], \end{aligned} \quad (3.38)$$

which we see is exactly in row-echelon form. This means the matrix is full rank, $N(K+1)$, since all the block-rows are linearly independent. This means there will always be a unique solution for $\hat{\mathbf{x}}$ provided that

$$\check{\mathbf{P}}_0 > 0, \quad \mathbf{Q}_k > 0, \quad (3.39)$$

where > 0 means a matrix is positive-definite (and hence invertible). The intuition behind this is that the prior already provides a complete solution to the problem. The measurements only serve to adjust the answer. Note that these are sufficient but not necessary conditions.

Case (ii): No knowledge of initial state

Each block-column of \mathbf{H}^T represents some piece of information that we have about the system. The first block-column represents our knowledge about the initial state. Thus, removing knowledge of the initial

¹² Follows from \mathbf{Q} and \mathbf{R} being real symmetric positive-definite.

state results in the rank test considering

$$\begin{aligned} & \text{rank } \mathbf{H}^T \\ = \text{rank } & \left[\begin{array}{c|c} -\mathbf{A}_0^T & \mathbf{C}_0^T \\ \hline \mathbf{1} & -\mathbf{A}_1^T \\ & \mathbf{1} & \ddots \\ & & \ddots & -\mathbf{A}_{K-1}^T \\ & & & \mathbf{1} \end{array} \right] , \end{aligned} \quad (3.40)$$

which we note has $K + 1$ block-rows (each of size N). Moving the top block-row to the bottom does not alter the rank:

$$\begin{aligned} & \text{rank } \mathbf{H}^T \\ = \text{rank } & \left[\begin{array}{c|c} \mathbf{1} & -\mathbf{A}_1^T \\ \mathbf{1} & \ddots \\ \ddots & -\mathbf{A}_{K-1}^T \\ \hline -\mathbf{A}_0^T & \mathbf{C}_0^T \end{array} \right] . \end{aligned} \quad (3.41)$$

Except for the bottom block-row, this is in row-echelon form. Again without altering the rank, we can add to the bottom block-row, \mathbf{A}_0^T times the first block-row, $\mathbf{A}_0^T \mathbf{A}_1^T$ times the second block-row, \dots , and $\mathbf{A}_0^T \cdots \mathbf{A}_{K-1}^T$ times the K th block-row, to see that

$$\begin{aligned} & \text{rank } \mathbf{H}^T \\ = \text{rank } & \left[\begin{array}{c|c} \mathbf{1} & -\mathbf{A}_1^T \\ \mathbf{1} & \ddots \\ \ddots & -\mathbf{A}_{K-1}^T \\ \hline \mathbf{C}_0^T & \mathbf{A}_0^T \mathbf{C}_1^T \\ & \mathbf{A}_0^T \mathbf{A}_1^T \mathbf{C}_2^T \\ & \cdots \\ & \mathbf{A}_0^T \cdots \mathbf{A}_{K-1}^T \mathbf{C}_K^T \end{array} \right] . \end{aligned} \quad (3.42)$$

Examining this last expression, we notice immediately that the lower-left partition is zero. Moreover, the upper-left partition is in row-echelon form and in fact is of full rank, NK , since every row has a ‘leading one’. Our overall rank condition for \mathbf{H}^T therefore collapses to showing that the lower-right partition has rank N :

$$\text{rank } [\mathbf{C}_0^T \ \mathbf{A}_0^T \mathbf{C}_1^T \ \mathbf{A}_0^T \mathbf{A}_1^T \mathbf{C}_2^T \ \cdots \ \mathbf{A}_0^T \cdots \mathbf{A}_{K-1}^T \mathbf{C}_K^T] = N. \quad (3.43)$$

If we further assume the system is time-invariant such that for all k we have $\mathbf{A}_k = \mathbf{A}$ and $\mathbf{C}_k = \mathbf{C}$ (we use italicized symbols to avoid confusion with lifted form) and we make the not-too-restrictive assumption that $K \gg N$, we may further simplify this condition.

To do so, we employ the *Cayley-Hamilton theorem* from linear algebra. Because \mathbf{A} is $N \times N$, its characteristic equation has at most N terms, and therefore any power of \mathbf{A} greater than or equal to N can be rewritten as a linear combination of $\mathbf{1}, \mathbf{A}, \dots, \mathbf{A}^{(N-1)}$. By extension, for any $k \geq N$, we can write

$$\begin{aligned} & (\mathbf{A}^T)^{(k-1)} \mathbf{C}^T \\ &= a_0 \mathbf{1}^T \mathbf{C}^T + a_1 \mathbf{A}^T \mathbf{C}^T + a_2 \mathbf{A}^T \mathbf{A}^T \mathbf{C}^T + \dots + a_{N-1} (\mathbf{A}^T)^{(N-1)} \mathbf{C}^T \end{aligned} \quad (3.44)$$

for some set of scalars, a_0, a_1, \dots, a_{N-1} , not all zero. Since row-rank and column-rank are the same for any matrix, we can conclude that

$$\begin{aligned} \text{rank } & \left[\mathbf{C}^T \ \mathbf{A}^T \mathbf{C}^T \ \mathbf{A}^T \mathbf{A}^T \mathbf{C}^T \ \dots \ (\mathbf{A}^T)^K \mathbf{C}^T \right] \\ &= \text{rank } \left[\mathbf{C}^T \ \mathbf{A}^T \mathbf{C}^T \ \dots \ (\mathbf{A}^T)^{(N-1)} \mathbf{C}^T \right]. \end{aligned} \quad (3.45)$$

Defining the *observability matrix*, \mathcal{O} , as

$$\mathcal{O} = \begin{bmatrix} \mathbf{C} \\ \mathbf{CA} \\ \vdots \\ \mathbf{CA}^{(N-1)} \end{bmatrix}, \quad (3.46)$$

our rank condition is

$$\text{rank } \mathcal{O} = N. \quad (3.47)$$

Readers familiar with linear control theory will recognize this as precisely the test for *observability* (Kalman, 1960a). Thus, we can see the direct connection between observability and invertibility of $\mathbf{H}^T \mathbf{W}^{-1} \mathbf{H}$. The overall conditions for existence and uniqueness of a solution to (3.34) are

$$\mathbf{Q}_k > 0, \quad \mathbf{R}_k > 0, \quad \text{rank } \mathcal{O} = N, \quad (3.48)$$

where > 0 means a matrix is positive-definite (and hence invertible). Again, these are sufficient but not necessary conditions.

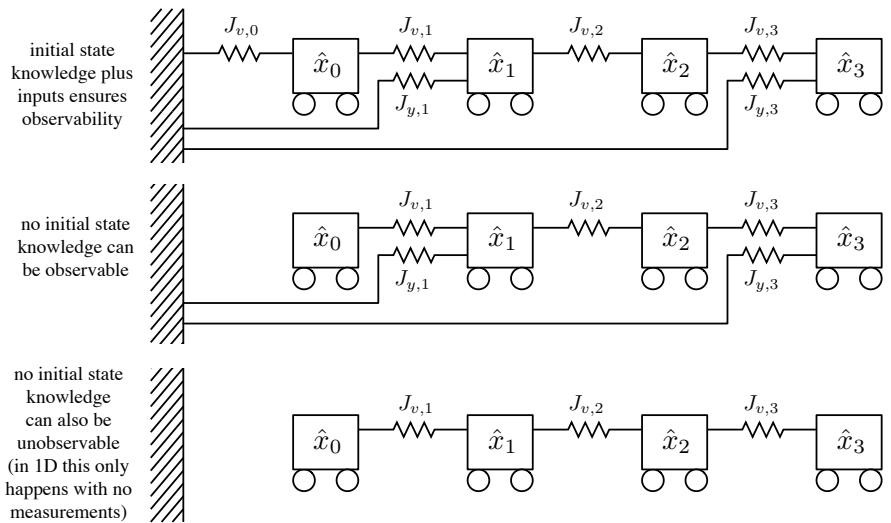
Cayley-Hamilton theorem: Every square matrix, \mathbf{A} , over the real field, satisfies its own characteristic equation,
 $\det(\lambda \mathbf{1} - \mathbf{A}) = 0$.

A system is *observable* if the initial state can be uniquely inferred based on measurements gathered in a finite amount of time.

Interpretation

We can return to the mass-spring analogy to better understand the observability issue. Figure 3.2 shows a few examples. With the initial state and all the inputs (top example), the system is always observable since it is impossible to move any group of carts left or right without altering the length of at least one spring. This means there is a unique minimum-energy state. The same is true for the middle example, even though there is no knowledge of the initial state. The bottom example

Figure 3.2 In a single dimension, the mass-spring system is observable if there is no group of carts that can be shifted left or right without altering the energy state of at least one spring. The top example uses the initial state and inputs, so this is always observable. The middle example is also observable since any movement changes at least one spring. The bottom example is not observable since the whole chain of carts can be moved left-right together without changing any spring lengths; in one dimension, this only happens with no initial state and no measurements.



is unobservable since the entire chain of carts can be moved left or right without changing the amount of energy stored in the springs. This means the minimum-energy state is not unique.

3.1.5 MAP Covariance

Looking back to (3.35), $\hat{\mathbf{x}}$ represents the most likely estimate of \mathbf{x} , the true state. One important question to ask, is how confident are we in $\hat{\mathbf{x}}$? It turns out we can re-interpret the least-squares solution as a Gaussian estimate for \mathbf{x} in the following way:

$$\underbrace{(\mathbf{H}^T \mathbf{W}^{-1} \mathbf{H})}_{\substack{\text{inverse} \\ \text{covariance}}} \underbrace{\hat{\mathbf{x}}}_{\substack{\text{mean}}} = \underbrace{\mathbf{H}^T \mathbf{W}^{-1} \mathbf{z}}_{\substack{\text{information} \\ \text{vector}}}. \quad (3.49)$$

The right-hand side is referred to as the *information vector*. To see this, we employ Bayes' rule to rewrite (3.15) as

$$p(\mathbf{x}|\mathbf{z}) = \beta \exp \left(-\frac{1}{2} (\mathbf{H}\mathbf{x} - \mathbf{z})^T \mathbf{W}^{-1} (\mathbf{H}\mathbf{x} - \mathbf{z}) \right), \quad (3.50)$$

where β is a new normalization constant. We then substitute (3.35) in and, after a little manipulation, find that

$$p(\mathbf{x}|\hat{\mathbf{x}}) = \kappa \exp \left(-\frac{1}{2} (\mathbf{x} - \hat{\mathbf{x}})^T (\mathbf{H}^T \mathbf{W}^{-1} \mathbf{H}) (\mathbf{x} - \hat{\mathbf{x}}) \right), \quad (3.51)$$

where κ is yet another normalization constant. We see from this that $\mathcal{N}(\hat{\mathbf{x}}, \hat{\mathbf{P}})$ is a Gaussian estimator for \mathbf{x} whose mean is the optimization solution and whose covariance is $\hat{\mathbf{P}} = (\mathbf{H}^T \mathbf{W}^{-1} \mathbf{H})^{-1}$.

Another way to explain this is to directly take the expectation of the estimate. We notice that

$$\mathbf{x} - \underbrace{(\mathbf{H}^T \mathbf{W}^{-1} \mathbf{H})^{-1} \mathbf{H}^T \mathbf{W}^{-1} \mathbf{z}}_{E[\mathbf{x}]} = (\mathbf{H}^T \mathbf{W}^{-1} \mathbf{H})^{-1} \mathbf{H}^T \mathbf{W}^{-1} \underbrace{(\mathbf{H}\mathbf{x} - \mathbf{z})}_{\mathbf{s}}, \quad (3.52)$$

where

$$\mathbf{s} = \begin{bmatrix} \mathbf{w} \\ \mathbf{n} \end{bmatrix}. \quad (3.53)$$

In this case we have

$$\begin{aligned} \hat{\mathbf{P}} &= E \left[(\mathbf{x} - E[\mathbf{x}]) (\mathbf{x} - E[\mathbf{x}])^T \right] \\ &= (\mathbf{H}^T \mathbf{W}^{-1} \mathbf{H})^{-1} \mathbf{H}^T \mathbf{W}^{-1} \underbrace{E[\mathbf{s}\mathbf{s}^T]}_{\mathbf{w}} \mathbf{W}^{-1} \mathbf{H} (\mathbf{H}^T \mathbf{W}^{-1} \mathbf{H})^{-1}, \\ &= (\mathbf{H}^T \mathbf{W}^{-1} \mathbf{H})^{-1}, \end{aligned} \quad (3.54)$$

which is the same result as above.

3.2 Recursive Discrete-Time Smoothing

The batch solution is appealing in that it is fairly easy to set up and understand from a least-squares perspective. However, brute-force solving the resulting system of linear equations will likely not be very efficient for most situations. Fortunately, since the inverse covariance matrix on the left-hand side is sparse (i.e., block-tridiagonal), we can use this to solve the system of equations very efficiently. This typically involves a forward recursion followed by a backward recursion. When the equations are solved in this way, the method is typically referred to as a *fixed-interval smoother*. It is useful to think of smoothers as efficiently implementing the full batch solution, with no approximation. We use the rest of this section to show that this can be done, first by a sparse Cholesky approach and then by the algebraically equivalent classical *Rauch-Tung-Striebel smoother* (Rauch et al., 1965). Särkkä (2013) provides an excellent reference on smoothing and filtering.

3.2.1 Exploiting Sparsity in the Batch Solution

As discussed earlier, the left-hand side of (3.34), $\mathbf{H}^T \mathbf{W}^{-1} \mathbf{H}$, is block-tridiagonal (under our chronological variable ordering for \mathbf{x}):

$$\mathbf{H}^T \mathbf{W}^{-1} \mathbf{H} = \begin{bmatrix} * & * & & & \\ * & * & * & & \\ * & * & * & & \\ & \ddots & \ddots & \ddots & \\ & & * & * & * \\ & & & * & * \end{bmatrix}, \quad (3.55)$$

André-Louis Cholesky (1875-1918) was a French military officer and mathematician. He is primarily remembered for a particular decomposition of matrices, which he used in his military map-making work. The Cholesky decomposition of a Hermitian positive-definite matrix, \mathbf{A} , is a decomposition of the form $\mathbf{A} = \mathbf{L}\mathbf{L}^*$ where \mathbf{L} is a lower-triangular matrix with real and positive diagonal entries and \mathbf{L}^* denotes the conjugate transpose of \mathbf{L} . Every Hermitian positive-definite matrix (and thus also every real-valued symmetric positive-definite matrix) has a unique Cholesky decomposition.

where $*$ indicates a non-zero block. There are solvers that can exploit this structure and therefore solve for $\hat{\mathbf{x}}$ efficiently.

One way to solve the batch equations efficiently is to do a sparse *Cholesky decomposition* followed by forward and backward passes. It turns out we can efficiently factor $\mathbf{H}^T \mathbf{W}^{-1} \mathbf{H}$ into

$$\mathbf{H}^T \mathbf{W}^{-1} \mathbf{H} = \mathbf{L}\mathbf{L}^T, \quad (3.56)$$

where \mathbf{L} is a block-lower-triangular matrix called the Cholesky factor¹³. Owing to the block-tridiagonal structure of $\mathbf{H}^T \mathbf{W}^{-1} \mathbf{H}$, \mathbf{L} will have the form,

$$\mathbf{L} = \begin{bmatrix} * & & & & \\ * & * & & & \\ * & * & * & & \\ & \ddots & \ddots & \ddots & \\ & & * & * & \\ & & & * & * \end{bmatrix}, \quad (3.57)$$

and the decomposition can be computed in $O(N(K + 1))$ time. Next, we solve

$$\mathbf{L}\mathbf{d} = \mathbf{H}^T \mathbf{W}^{-1} \mathbf{z}, \quad (3.58)$$

for \mathbf{d} . This can again be done in $O(N(K + 1))$ time through forward substitution owing to the sparse lower-triangular form of \mathbf{L} ; this is called the forward pass. Finally, we solve

$$\mathbf{L}^T \hat{\mathbf{x}} = \mathbf{d}, \quad (3.59)$$

for $\hat{\mathbf{x}}$, which again can be done in $O(N(K + 1))$ time through backward substitution owing to the sparse upper-triangular form of \mathbf{L}^T ; this is called the backward pass. Thus, the batch equations can be solved in computation time that scales linearly with the size of the state. The next section will make the details of this sparse Cholesky approach specific.

¹³ We could just as easily factor into $\mathbf{H}^T \mathbf{W}^{-1} \mathbf{H} = \mathbf{U}\mathbf{U}^T$ with \mathbf{U} upper-triangular and then carry out backward and forward passes.

3.2.2 Cholesky Smoother

In this section, we work out the details of the sparse Cholesky solution to the batch estimation problem. The result will be a set of forward-backward recursions that we will refer to as the *Cholesky smoother*. There are several similar square-root information smoothers described in the literature, and Bierman (1974) is a classic reference on the topic.

Let us begin by defining the non-zero sub-blocks of \mathbf{L} as

$$\mathbf{L} = \begin{bmatrix} \mathbf{L}_0 & & & & \\ \mathbf{L}_{10} & \mathbf{L}_1 & & & \\ & \mathbf{L}_{21} & \mathbf{L}_2 & & \\ & & \ddots & \ddots & \\ & & & \mathbf{L}_{K-1,K-2} & \mathbf{L}_{K-1} \\ & & & & \mathbf{L}_{K,K-1} & \mathbf{L}_K \end{bmatrix}. \quad (3.60)$$

Using the definitions of \mathbf{H} and \mathbf{W} from (3.13), when we multiply out $\mathbf{H}^T \mathbf{W}^{-1} \mathbf{H} = \mathbf{L} \mathbf{L}^T$ and compare at the block level, we have

$$\mathbf{L}_0 \mathbf{L}_0^T = \underbrace{\check{\mathbf{P}}_0^{-1} + \mathbf{C}_0^T \mathbf{R}_0^{-1} \mathbf{C}_0}_{\mathbf{I}_0} + \mathbf{A}_0^T \mathbf{Q}_1^{-1} \mathbf{A}_0, \quad (3.61a)$$

$$\mathbf{L}_{10} \mathbf{L}_{10}^T = -\mathbf{Q}_1^{-1} \mathbf{A}_0, \quad (3.61b)$$

$$\mathbf{L}_1 \mathbf{L}_1^T = \underbrace{-\mathbf{L}_{10} \mathbf{L}_{10}^T + \mathbf{Q}_1^{-1}}_{\mathbf{I}_1} + \mathbf{C}_1^T \mathbf{R}_1^{-1} \mathbf{C}_1 + \mathbf{A}_1^T \mathbf{Q}_2^{-1} \mathbf{A}_1, \quad (3.61c)$$

$$\mathbf{L}_{21} \mathbf{L}_{21}^T = -\mathbf{Q}_2^{-1} \mathbf{A}_1, \quad (3.61d)$$

⋮

$$\mathbf{L}_{K-1} \mathbf{L}_{K-1}^T = \underbrace{-\mathbf{L}_{K-1,K-2} \mathbf{L}_{K-1,K-2}^T + \mathbf{Q}_{K-1}^{-1} + \mathbf{C}_{K-1}^T \mathbf{R}_{K-1}^{-1} \mathbf{C}_{K-1}}_{\mathbf{I}_{K-1}} + \mathbf{A}_{K-1}^T \mathbf{Q}_K^{-1} \mathbf{A}_{K-1}, \quad (3.61e)$$

$$\mathbf{L}_{K,K-1} \mathbf{L}_{K,K-1}^T = -\mathbf{Q}_K^{-1} \mathbf{A}_{K-1}, \quad (3.61f)$$

$$\mathbf{L}_K \mathbf{L}_K^T = \underbrace{-\mathbf{L}_{K,K-1} \mathbf{L}_{K,K-1}^T + \mathbf{Q}_K^{-1} + \mathbf{C}_K^T \mathbf{R}_K^{-1} \mathbf{C}_K}_{\mathbf{I}_K}, \quad (3.61g)$$

where the underbraces allow us to define the \mathbf{I}_k quantities¹⁴, whose purpose will be revealed shortly. From these equations, we can first solve for \mathbf{L}_0 by doing a small (dense) Cholesky decomposition in the first equation, then substitute this into the second to solve for \mathbf{L}_{10} , then substitute this into the third to solve for \mathbf{L}_1 , and so on all the way down to \mathbf{L}_K . This confirms that we can work out all the blocks of \mathbf{L} in a single forwards pass in $O(N(K + 1))$ time.

¹⁴ In this book, $\mathbf{1}$ is the identity matrix, which should not be confused with the use of \mathbf{I} , which in this instance stands for *information matrix* (i.e., inverse covariance matrix).

Next we solve $\mathbf{L}\mathbf{d} = \mathbf{H}^T\mathbf{W}^{-1}\mathbf{z}$ for \mathbf{d} , where

$$\mathbf{d} = \begin{bmatrix} \mathbf{d}_0 \\ \mathbf{d}_1 \\ \vdots \\ \mathbf{d}_K \end{bmatrix}. \quad (3.62)$$

Multiplying out and comparing at the block level, we have

$$\mathbf{L}_0\mathbf{d}_0 = \underbrace{\check{\mathbf{P}}_0^{-1}\check{\mathbf{x}}_0 + \mathbf{C}_0^T\mathbf{R}_0^{-1}\mathbf{y}_0}_{\mathbf{q}_0} - \mathbf{A}_0^T\mathbf{Q}_1^{-1}\mathbf{v}_1, \quad (3.63a)$$

$$\mathbf{L}_1\mathbf{d}_1 = \underbrace{-\mathbf{L}_{10}\mathbf{d}_0 + \mathbf{Q}_1^{-1}\mathbf{v}_1 + \mathbf{C}_1^T\mathbf{R}_1^{-1}\mathbf{y}_1}_{\mathbf{q}_1} - \mathbf{A}_1^T\mathbf{Q}_2^{-1}\mathbf{v}_2, \quad (3.63b)$$

⋮

$$\mathbf{L}_{K-1}\mathbf{d}_{K-1} = \underbrace{-\mathbf{L}_{K-1,K-2}\mathbf{d}_{K-2} + \mathbf{Q}_{K-1}^{-1}\mathbf{v}_{K-1} + \mathbf{C}_{K-1}^T\mathbf{R}_{K-1}^{-1}\mathbf{y}_{K-1}}_{\mathbf{q}_{K-1}} - \mathbf{A}_{K-1}^T\mathbf{Q}_K^{-1}\mathbf{v}_K, \quad (3.63c)$$

$$\mathbf{L}_K\mathbf{d}_K = \underbrace{-\mathbf{L}_{K,K-1}\mathbf{d}_{K-1} + \mathbf{Q}_K^{-1}\mathbf{v}_K + \mathbf{C}_K^T\mathbf{R}_K^{-1}\mathbf{y}_K}_{\mathbf{q}_K}, \quad (3.63d)$$

where again the underbraces allow us to define the \mathbf{q}_k quantities, which will be used shortly. From these equations, we can solve for \mathbf{d}_0 in the first equation, then substitute this into the second to solve for \mathbf{d}_1 , and so on all the way down to \mathbf{d}_K . This confirms that we can work out all of the blocks of \mathbf{d} in a single forward pass in $O(N(K+1))$ time.

The last step in the Cholesky approach is to solve $\mathbf{L}^T\hat{\mathbf{x}} = \mathbf{d}$ for $\hat{\mathbf{x}}$, where

$$\hat{\mathbf{x}} = \begin{bmatrix} \hat{\mathbf{x}}_0 \\ \hat{\mathbf{x}}_1 \\ \vdots \\ \hat{\mathbf{x}}_K \end{bmatrix}. \quad (3.64)$$

Multiplying out and comparing at the block level, we have

$$\mathbf{L}_K^T\hat{\mathbf{x}}_K = \mathbf{d}_K, \quad (3.65a)$$

$$\mathbf{L}_{K-1}^T\hat{\mathbf{x}}_{K-1} = -\mathbf{L}_{K,K-1}^T\hat{\mathbf{x}}_K + \mathbf{d}_{K-1}, \quad (3.65b)$$

⋮

$$\mathbf{L}_1^T\hat{\mathbf{x}}_1 = -\mathbf{L}_{21}^T\hat{\mathbf{x}}_2 + \mathbf{d}_1, \quad (3.65c)$$

$$\mathbf{L}_0^T\hat{\mathbf{x}}_0 = -\mathbf{L}_{10}^T\hat{\mathbf{x}}_1 + \mathbf{d}_0. \quad (3.65d)$$

From these equations, we can solve for $\hat{\mathbf{x}}_K$ in the first equation, then substitute this into the second to solve for $\hat{\mathbf{x}}_{K-1}$, and so on all the way down to $\hat{\mathbf{x}}_0$. This confirms that we can work out all of the blocks of $\hat{\mathbf{x}}$ in a single backward pass in $O(N(K+1))$ time.

In terms of the \mathbf{I}_k and \mathbf{q}_k quantities, we can combine the two forwards passes (to solve for \mathbf{L} and \mathbf{d}) and also write the backwards pass as

forward:

$$(k = 1 \dots K)$$

$$\mathbf{L}_{k-1} \mathbf{L}_{k-1}^T = \mathbf{I}_{k-1} + \mathbf{A}_{k-1}^T \mathbf{Q}_k^{-1} \mathbf{A}_{k-1}, \quad (3.66a)$$

$$\mathbf{L}_{k-1} \mathbf{d}_{k-1} = \mathbf{q}_{k-1} - \mathbf{A}_{k-1}^T \mathbf{Q}_k^{-1} \mathbf{v}_k, \quad (3.66b)$$

$$\mathbf{L}_{k,k-1} \mathbf{L}_{k-1}^T = -\mathbf{Q}_k^{-1} \mathbf{A}_{k-1}, \quad (3.66c)$$

$$\mathbf{I}_k = -\mathbf{L}_{k,k-1} \mathbf{L}_{k,k-1}^T + \mathbf{Q}_k^{-1} + \mathbf{C}_k^T \mathbf{R}_k^{-1} \mathbf{C}_k, \quad (3.66d)$$

$$\mathbf{q}_k = -\mathbf{L}_{k,k-1} \mathbf{d}_{k-1} + \mathbf{Q}_k^{-1} \mathbf{v}_k + \mathbf{C}_k^T \mathbf{R}_k^{-1} \mathbf{y}_k, \quad (3.66e)$$

backward:

$$(k = K \dots 1)$$

$$\mathbf{L}_{k-1}^T \hat{\mathbf{x}}_{k-1} = -\mathbf{L}_{k,k-1}^T \hat{\mathbf{x}}_k + \mathbf{d}_{k-1}, \quad (3.66f)$$

which are initialized with

$$\mathbf{I}_0 = \check{\mathbf{P}}_0^{-1} + \mathbf{C}_0^T \mathbf{R}_0^{-1} \mathbf{C}_0, \quad (3.67a)$$

$$\mathbf{q}_0 = \check{\mathbf{P}}_0^{-1} \check{\mathbf{x}}_0 + \mathbf{C}_0^T \mathbf{R}_0^{-1} \mathbf{y}_0, \quad (3.67b)$$

$$\hat{\mathbf{x}}_K = \mathbf{L}_K^{-T} \mathbf{d}_K. \quad (3.67c)$$

The forward pass maps $\{\mathbf{q}_{k-1}, \mathbf{I}_{k-1}\}$ to the same pair at the next time, $\{\mathbf{q}_k, \mathbf{I}_k\}$. The backward pass maps $\hat{\mathbf{x}}_k$ to the same quantity at the previous timestep, $\hat{\mathbf{x}}_{k-1}$. In the process, we solve for all the blocks of \mathbf{L} and \mathbf{d} . The only linear algebra operations required to implement this smoother are Cholesky decomposition, multiplication, addition, and solving a linear system via forward/backward substitution.

As we will see in the next section, these six recursive equations are algebraically equivalent to the canonical *Rauch-Tung-Striebel smoother*, the five equations forming the forward pass are algebraically equivalent to the famous *Kalman filter*.

Herbert E. Rauch (1935-2011) was a pioneer in the area of control and estimation. Frank F. Tung (1933-2006) was a research scientist working in the area of computing and control. Charlotte T. Striebel (1929-2014) was a statistician and professor of mathematics. All three co-developed the Rauch-Tung-Striebel smoother while working at Lockheed Missiles and Space Company in order to estimate spacecraft trajectories.

3.2.3 Rauch-Tung-Striebel Smoother

While the Cholesky smoother is a convenient implementation and is easy to understand when starting from the batch solution, it does not represent the canonical form of the smoothing equations. It is, however, algebraically equivalent to the canonical Rauch-Tung-Striebel (RTS) smoother, which we now show. This requires several uses of the different forms of the SMW identity in (2.75).

We begin by working on the forward pass. Solving for $\mathbf{L}_{k,k-1}$ in (3.66c)

and substituting this and (3.66a) into (3.66d), we have

$$\mathbf{I}_k = \underbrace{\mathbf{Q}_k^{-1} - \mathbf{Q}_k^{-1} \mathbf{A}_{k-1} \left(\mathbf{I}_{k-1} + \mathbf{A}_{k-1}^T \mathbf{Q}_k^{-1} \mathbf{A}_{k-1} \right)^{-1} \mathbf{A}_{k-1}^T \mathbf{Q}_k^{-1}}_{\left(\mathbf{A}_{k-1} \mathbf{I}_{k-1}^{-1} \mathbf{A}_{k-1}^T + \mathbf{Q}_k \right)^{-1}, \text{ by (2.75)}} + \mathbf{C}_k^T \mathbf{R}_k^{-1} \mathbf{C}_k \quad (3.68)$$

where we have used a version of the SMW identity to get to the expression in the underbrace. By letting $\hat{\mathbf{P}}_{k,f} = \mathbf{I}_k^{-1}$, this can be written in two steps as

$$\check{\mathbf{P}}_{k,f} = \mathbf{A}_{k-1} \hat{\mathbf{P}}_{k-1,f} \mathbf{A}_{k-1}^T + \mathbf{Q}_k, \quad (3.69a)$$

$$\hat{\mathbf{P}}_{k,f}^{-1} = \check{\mathbf{P}}_{k,f}^{-1} + \mathbf{C}_k^T \mathbf{R}_k^{-1} \mathbf{C}_k, \quad (3.69b)$$

where $\check{\mathbf{P}}_{k,f}$ represents a ‘predicted’ covariance and $\hat{\mathbf{P}}_{k,f}$ a ‘corrected’ one. We have added the subscript, $(\cdot)_f$, to indicate these quantities come from the forward pass (i.e., a filter). The second of these equations is written in *information* (i.e., inverse covariance) form. To reach the canonical version, we define the *Kalman gain matrix*, \mathbf{K}_k , as

$$\mathbf{K}_k = \hat{\mathbf{P}}_{k,f} \mathbf{C}_k^T \mathbf{R}_k^{-1}. \quad (3.70)$$

Substituting in (3.69b), this can also be written as

$$\begin{aligned} \mathbf{K}_k &= (\check{\mathbf{P}}_{k,f}^{-1} + \mathbf{C}_k^T \mathbf{R}_k^{-1} \mathbf{C}_k)^{-1} \mathbf{C}_k^T \mathbf{R}_k^{-1} \\ &= \check{\mathbf{P}}_{k,f} \mathbf{C}_k^T (\mathbf{C}_k \check{\mathbf{P}}_{k,f} \mathbf{C}_k^T + \mathbf{R}_k)^{-1}, \end{aligned} \quad (3.71)$$

where the last expression requires a use of the SMW identity from (2.75). Then (3.69b) can be rewritten as

$$\begin{aligned} \check{\mathbf{P}}_{k,f}^{-1} &= \hat{\mathbf{P}}_{k,f}^{-1} - \mathbf{C}_k^T \mathbf{R}_k^{-1} \mathbf{C}_k = \hat{\mathbf{P}}_{k,f}^{-1} \left(\mathbf{1} - \underbrace{\hat{\mathbf{P}}_{k,f} \mathbf{C}_k^T \mathbf{R}_k^{-1} \mathbf{C}_k}_{\mathbf{K}_k} \right) \\ &= \hat{\mathbf{P}}_{k,f}^{-1} (\mathbf{1} - \mathbf{K}_k \mathbf{C}_k), \end{aligned} \quad (3.72)$$

and finally, rearranging for $\hat{\mathbf{P}}_{k,f}$, we have

$$\hat{\mathbf{P}}_{k,f} = (\mathbf{1} - \mathbf{K}_k \mathbf{C}_k) \check{\mathbf{P}}_{k,f}, \quad (3.73)$$

which is the canonical form for the covariance correction step.

Next, solving for $\mathbf{L}_{k,k-1}$ in (3.66c) and \mathbf{d}_{k-1} in (3.66b), we have

$$\mathbf{L}_{k,k-1} \mathbf{d}_{k-1} = -\mathbf{Q}_k^{-1} \mathbf{A}_{k-1} \left(\mathbf{L}_{k-1} \mathbf{L}_{k-1}^T \right)^{-1} \left(\mathbf{q}_{k-1} - \mathbf{A}_{k-1}^T \mathbf{Q}_k^{-1} \mathbf{v}_k \right). \quad (3.74)$$

Substituting (3.66a) into $\mathbf{L}_{k,k-1}\mathbf{d}_{k-1}$ and then this into (3.66e), we have

$$\begin{aligned} \mathbf{q}_k = & \underbrace{\mathbf{Q}_k^{-1}\mathbf{A}_{k-1}\left(\mathbf{I}_{k-1} + \mathbf{A}_{k-1}^T\mathbf{Q}_k^{-1}\mathbf{A}_{k-1}\right)^{-1}}_{\left(\mathbf{A}_{k-1}\mathbf{I}_{k-1}^{-1}\mathbf{A}_{k-1}^T + \mathbf{Q}_k\right)^{-1}\mathbf{A}_{k-1}\mathbf{I}_{k-1}^{-1}, \text{ by (2.75)}} \mathbf{q}_{k-1} \\ & + \underbrace{\left(\mathbf{Q}_k^{-1} - \mathbf{Q}_k^{-1}\mathbf{A}_{k-1}\left(\mathbf{I}_{k-1} + \mathbf{A}_{k-1}^T\mathbf{Q}_k^{-1}\mathbf{A}_{k-1}\right)^{-1}\mathbf{A}_{k-1}^T\mathbf{Q}_k^{-1}\right)\mathbf{v}_k}_{\left(\mathbf{A}_{k-1}\mathbf{I}_{k-1}^{-1}\mathbf{A}_{k-1}^T + \mathbf{Q}_k\right)^{-1}, \text{ by (2.75)}} \\ & + \mathbf{C}_k^T\mathbf{R}_k^{-1}\mathbf{y}_k, \quad (3.75) \end{aligned}$$

where we have used two versions of the SMW identity to get to the expressions in the underbraces. By letting $\hat{\mathbf{P}}_{k,f}^{-1}\hat{\mathbf{x}}_{k,f} = \mathbf{q}_k$, this can be written in two steps as

$$\check{\mathbf{x}}_{k,f} = \mathbf{A}_{k-1}\hat{\mathbf{x}}_{k-1,f} + \mathbf{v}_k, \quad (3.76a)$$

$$\hat{\mathbf{P}}_{k,f}^{-1}\hat{\mathbf{x}}_{k,f} = \check{\mathbf{x}}_{k,f} + \mathbf{C}_k^T\mathbf{R}_k^{-1}\mathbf{y}_k, \quad (3.76b)$$

where $\check{\mathbf{x}}_{k,f}$ represents a ‘predicted’ mean and $\hat{\mathbf{x}}_{k,f}$ a ‘corrected’ one. Again, the second of these is in *information* (i.e., inverse covariance) form. To get to the canonical form, we rewrite it as

$$\hat{\mathbf{x}}_{k,f} = \underbrace{\hat{\mathbf{P}}_{k,f}\check{\mathbf{x}}_{k,f}}_{\mathbf{1} - \mathbf{K}_k\mathbf{C}_k} + \underbrace{\hat{\mathbf{P}}_{k,f}\mathbf{C}_k^T\mathbf{R}_k^{-1}}_{\mathbf{K}_k}\mathbf{y}_k, \quad (3.77)$$

or

$$\hat{\mathbf{x}}_{k,f} = \check{\mathbf{x}}_{k,f} + \mathbf{K}_k(\mathbf{y}_k - \mathbf{C}_k\check{\mathbf{x}}_{k,f}), \quad (3.78)$$

which is the canonical form for the mean correction step.

The last step is to resolve the backward pass into its canonical form. We begin by premultiplying (3.66f) by \mathbf{L}_{k-1} and solving for $\hat{\mathbf{x}}_{k-1}$:

$$\hat{\mathbf{x}}_{k-1} = (\mathbf{L}_{k-1}\mathbf{L}_{k-1}^T)^{-1}\mathbf{L}_{k-1}(-\mathbf{L}_{k-1}^T\hat{\mathbf{x}}_k + \mathbf{d}_{k-1}). \quad (3.79)$$

Substituting in (3.66a), (3.66b), and (3.66c), we have

$$\begin{aligned} \hat{\mathbf{x}}_{k-1} = & \underbrace{\left(\mathbf{I}_{k-1} + \mathbf{A}_{k-1}^T\mathbf{Q}_k^{-1}\mathbf{A}_{k-1}\right)^{-1}\mathbf{A}_{k-1}^T\mathbf{Q}_k^{-1}}_{\mathbf{I}_{k-1}^{-1}\mathbf{A}_{k-1}^T\left(\mathbf{A}_{k-1}\mathbf{I}_{k-1}^{-1}\mathbf{A}_{k-1}^T + \mathbf{Q}_k\right)^{-1}, \text{ by (2.75)}} (\hat{\mathbf{x}}_k - \mathbf{v}_k) \\ & + \underbrace{\left(\mathbf{I}_{k-1} + \mathbf{A}_{k-1}^T\mathbf{Q}_k^{-1}\mathbf{A}_{k-1}\right)^{-1}}_{\mathbf{I}_{k-1}^{-1}\mathbf{A}_{k-1}^T\left(\mathbf{A}_{k-1}\mathbf{I}_{k-1}^{-1}\mathbf{A}_{k-1}^T + \mathbf{Q}_k\right)^{-1}\mathbf{A}_{k-1}\mathbf{I}_{k-1}^{-1}, \text{ by (2.75)}} \mathbf{q}_{k-1}. \quad (3.80) \end{aligned}$$

Using our symbols from above, this can be written as

$$\hat{\mathbf{x}}_{k-1} = \hat{\mathbf{x}}_{k-1,f} + \hat{\mathbf{P}}_{k-1,f}\mathbf{A}_{k-1}^T\hat{\mathbf{P}}_{k,f}^{-1}(\hat{\mathbf{x}}_k - \check{\mathbf{x}}_{k,f}), \quad (3.81)$$

which is the canonical form for the backward smoothing equation.

Together, equations (3.69a), (3.71), (3.73), (3.76a), (3.78), and (3.81) constitute the Rauch-Tung-Striebel smoother:

forward:

$$(k = 1 \dots K)$$

$$\check{\mathbf{P}}_{k,f} = \mathbf{A}_{k-1} \hat{\mathbf{P}}_{k-1,f} \mathbf{A}_{k-1}^T + \mathbf{Q}_k, \quad (3.82a)$$

$$\check{\mathbf{x}}_{k,f} = \mathbf{A}_{k-1} \hat{\mathbf{x}}_{k-1,f} + \mathbf{v}_k, \quad (3.82b)$$

$$\mathbf{K}_k = \check{\mathbf{P}}_{k,f} \mathbf{C}_k^T (\mathbf{C}_k \check{\mathbf{P}}_{k,f} \mathbf{C}_k^T + \mathbf{R}_k)^{-1}, \quad (3.82c)$$

$$\hat{\mathbf{P}}_{k,f} = (\mathbf{I} - \mathbf{K}_k \mathbf{C}_k) \check{\mathbf{P}}_{k,f}, \quad (3.82d)$$

$$\hat{\mathbf{x}}_{k,f} = \check{\mathbf{x}}_{k,f} + \mathbf{K}_k (\mathbf{y}_k - \mathbf{C}_k \check{\mathbf{x}}_{k,f}), \quad (3.82e)$$

backward:

$$(k = K \dots 1)$$

$$\hat{\mathbf{x}}_{k-1} = \hat{\mathbf{x}}_{k-1,f} + (\hat{\mathbf{P}}_{k-1,f} \mathbf{A}_{k-1}^T \check{\mathbf{P}}_{k,f}^{-1}) (\hat{\mathbf{x}}_k - \check{\mathbf{x}}_{k,f}), \quad (3.82f)$$

which are initialized with

$$\check{\mathbf{P}}_{0,f} = \check{\mathbf{P}}_0, \quad (3.83a)$$

$$\check{\mathbf{x}}_{0,f} = \check{\mathbf{x}}_0, \quad (3.83b)$$

$$\hat{\mathbf{x}}_K = \hat{\mathbf{x}}_{K,f}. \quad (3.83c)$$

As will be discussed in more detail in the next section, the five equations in the forward pass are known as the *Kalman filter*. However, the important message to take away from this section on smoothing is that these six equations representing the RTS smoother can be used to solve the original batch problem that we set up in a very efficient manner, with no approximation. This is possible precisely because of the block-tridiagonal sparsity pattern in the left-hand side of the batch problem.

3.3 Recursive Discrete-Time Filtering

The batch solution (and the corresponding smoother implementations) outlined above is really the best we can do. It makes use of all the data in the estimate of every state. However, it has one major drawback: it cannot be used online¹⁵ because it employs future data to estimate past states (i.e., it is not *causal*). To be used online, the estimate of the current state can only employ data up to the current timestep. The *Kalman filter* is the classical solution to this problem. We have already seen a preview of the KF; it is the forward pass of the Rauch-Tung-Striebel smoother. However, there are several other ways of deriving it, some of which we provide in this section.

¹⁵ It is preferable to say ‘online’ rather than ‘real-time’ in this context.

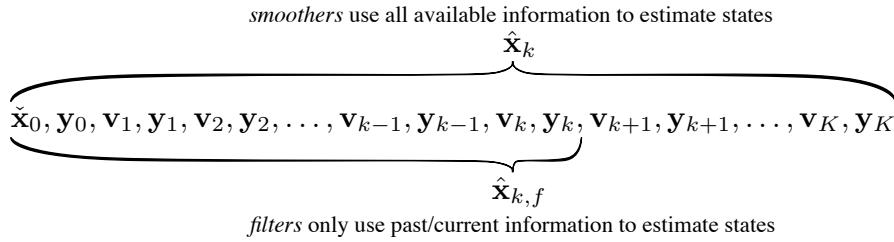


Figure 3.3 The batch LG solution is a *smoother*. To develop an estimator appropriate to online estimation, we require a *filter*.

3.3.1 Factoring the Batch Solution

We do not need to start from scratch in our search for a recursive LG estimator. It turns out we can re-use the batch solution and exactly factor it into two recursive estimators, one that runs forward in time and the other backward. The backward pass is a little different than the one presented in the smoother section, as it is not correcting the forward pass, but rather producing an estimate using only future measurements.

To set things up for our development of the recursive solutions, we will reorder some of our variables from the batch solution. We redefine \mathbf{z} , \mathbf{H} , and \mathbf{W} as

$$\mathbf{z} = \begin{bmatrix} \check{\mathbf{x}}_0 \\ \mathbf{y}_0 \\ \hline \mathbf{v}_1 \\ \mathbf{y}_1 \\ \hline \mathbf{v}_2 \\ \mathbf{y}_2 \\ \hline \vdots \\ \hline \mathbf{v}_K \\ \mathbf{y}_K \end{bmatrix}, \quad \mathbf{H} = \begin{bmatrix} 1 \\ \mathbf{C}_0 \\ \hline -\mathbf{A}_0 & 1 \\ \mathbf{C}_1 \\ \hline -\mathbf{A}_1 & 1 \\ \mathbf{C}_2 \\ \hline \ddots & \ddots \\ \hline -\mathbf{A}_{K-1} & 1 \\ \mathbf{C}_K \end{bmatrix},$$

$$\mathbf{W} = \left[\begin{array}{c|c|c|c|c} \check{\mathbf{P}}_0 & & & & \\ \mathbf{R}_0 & & & & \\ \hline & \mathbf{Q}_1 & & & \\ & \mathbf{R}_1 & & & \\ \hline & & \mathbf{Q}_2 & & \\ & & \mathbf{R}_2 & & \\ \hline & & & \ddots & \\ \hline & & & & \mathbf{Q}_K \\ & & & & \mathbf{R}_K \end{array} \right], \quad (3.84)$$

where the partition lines now show divisions between timesteps. This re-ordering does not change the ordering of \mathbf{x} , so $\mathbf{H}^T \mathbf{W}^{-1} \mathbf{H}$ is still block-tridiagonal.

We now consider the factorization at the probability density level. As discussed in Section 3.1.5, we have an expression for $p(\mathbf{x}|\mathbf{v}, \mathbf{y})$. If we

want to consider only the state at time k , we can marginalize out the other states by integrating over all possible values:

$$p(\mathbf{x}_k | \mathbf{v}, \mathbf{y}) = \int_{\mathbf{x}_{i,\forall i \neq k}} p(\mathbf{x}_0, \dots, \mathbf{x}_K | \mathbf{v}, \mathbf{y}) d\mathbf{x}_{i,\forall i \neq k}. \quad (3.85)$$

It turns out that we can factor this probability density into two parts:

$$p(\mathbf{x}_k | \mathbf{v}, \mathbf{y}) = \eta p(\mathbf{x}_k | \check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k}) p(\mathbf{x}_k | \mathbf{v}_{k+1:K}, \mathbf{y}_{k+1:K}), \quad (3.86)$$

where η is a normalization constant to enforce the axiom of total probability. In other words, we can take our batch solution and factor it into the normalized product of two Gaussian PDFs, as was discussed in Section 2.2.6.

To carry out this factorization, we exploit the sparse structure of \mathbf{H} in (3.3.1). We begin by partitioning \mathbf{H} into 12 blocks (only 6 of which are non-zero):

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_{11} & & & \\ \mathbf{H}_{21} & \mathbf{H}_{22} & & \\ & \mathbf{H}_{32} & \mathbf{H}_{33} & \\ & & \mathbf{H}_{43} & \\ \uparrow & \uparrow & \uparrow & \\ & & \text{states from } k+1 \dots K & \\ & & \uparrow & \\ & & \text{states from } k & \\ & & \uparrow & \\ & & \text{states from } 0 \dots k-1 & \end{bmatrix} \begin{array}{l} \text{information from } 0 \dots k-1 \\ \text{information from } k \\ \text{information from } k+1 \\ \text{information from } k+2 \dots K \end{array} \quad (3.87)$$

The sizes of each block-row and block-column are indicated above. For example, with $k = 2$ and $K = 4$, the partitions are

$$\mathbf{H} = \left[\begin{array}{c|c|c} \mathbf{1} & & \\ \mathbf{C}_0 & & \\ -\mathbf{A}_0 & \mathbf{1} & \\ & \mathbf{C}_1 & \\ \hline & -\mathbf{A}_1 & \mathbf{1} \\ & & \mathbf{C}_2 \\ \hline & -\mathbf{A}_2 & \mathbf{1} \\ & & \mathbf{C}_3 \\ \hline & & -\mathbf{A}_3 & \mathbf{1} \\ & & & \mathbf{C}_4 \end{array} \right]. \quad (3.88)$$

We use compatible partitions for \mathbf{z} and \mathbf{W} :

$$\mathbf{z} = \begin{bmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \\ \mathbf{z}_3 \\ \mathbf{z}_4 \end{bmatrix}, \quad \mathbf{W} = \begin{bmatrix} \mathbf{W}_1 & & & \\ & \mathbf{W}_2 & & \\ & & \mathbf{W}_3 & \\ & & & \mathbf{W}_4 \end{bmatrix}. \quad (3.89)$$

For $\mathbf{H}^T \mathbf{W}^{-1} \mathbf{H}$ we then have

$$\begin{aligned} & \mathbf{H}^T \mathbf{W}^{-1} \mathbf{H} \\ = & \begin{bmatrix} \mathbf{H}_{11}^T \mathbf{W}_1^{-1} \mathbf{H}_{11} + \mathbf{H}_{21}^T \mathbf{W}_2^{-1} \mathbf{H}_{21} & \mathbf{H}_{21}^T \mathbf{W}_2^{-1} \mathbf{H}_{22} \\ \mathbf{H}_{22}^T \mathbf{W}_2^{-1} \mathbf{H}_{21} & \mathbf{H}_{22}^T \mathbf{W}_2^{-1} \mathbf{H}_{22} + \mathbf{H}_{32}^T \mathbf{W}_3^{-1} \mathbf{H}_{32} \\ & \mathbf{H}_{33}^T \mathbf{W}_3^{-1} \mathbf{H}_{32} \end{bmatrix} \\ & \quad \cdots \quad \begin{bmatrix} \mathbf{H}_{32}^T \mathbf{W}_3^{-1} \mathbf{H}_{33} \\ \mathbf{H}_{33}^T \mathbf{W}_3^{-1} \mathbf{H}_{33} + \mathbf{H}_{43}^T \mathbf{W}_4^{-1} \mathbf{H}_{43} \end{bmatrix} \\ = & \begin{bmatrix} \mathbf{L}_{11} & \mathbf{L}_{12} & \mathbf{L}_{12}^T \\ \mathbf{L}_{12}^T & \mathbf{L}_{22} & \mathbf{L}_{32}^T \\ \mathbf{L}_{32} & \mathbf{L}_{33} & \mathbf{L}_{33} \end{bmatrix}, \quad (3.90) \end{aligned}$$

where we have assigned the blocks to some useful intermediate variables, \mathbf{L}_{ij} . For $\mathbf{H}^T \mathbf{W}^{-1} \mathbf{z}$ we have

$$\mathbf{H}^T \mathbf{W}^{-1} \mathbf{z} = \begin{bmatrix} \mathbf{H}_{11}^T \mathbf{W}_1^{-1} \mathbf{z}_1 + \mathbf{H}_{21}^T \mathbf{W}_2^{-1} \mathbf{z}_2 \\ \mathbf{H}_{22}^T \mathbf{W}_2^{-1} \mathbf{z}_2 + \mathbf{H}_{32}^T \mathbf{W}_3^{-1} \mathbf{z}_3 \\ \mathbf{H}_{33}^T \mathbf{W}_3^{-1} \mathbf{z}_3 + \mathbf{H}_{43}^T \mathbf{W}_4^{-1} \mathbf{z}_4 \end{bmatrix} = \begin{bmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \\ \mathbf{r}_3 \end{bmatrix}, \quad (3.91)$$

where we have assigned the blocks to some useful intermediate variables, \mathbf{r}_i . Next, we partition the states, \mathbf{x} , in the following way:

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_{0:k-1} \\ \mathbf{x}_k \\ \mathbf{x}_{k+1:K} \end{bmatrix} \begin{array}{l} \text{states from } 0 \dots k-1 \\ \text{states from } k \\ \text{states from } k+1 \dots K \end{array} \quad (3.92)$$

Our overall batch system of equations now looks like the following:

$$\begin{bmatrix} \mathbf{L}_{11} & \mathbf{L}_{12} & \mathbf{L}_{12}^T \\ \mathbf{L}_{12}^T & \mathbf{L}_{22} & \mathbf{L}_{32}^T \\ \mathbf{L}_{32} & \mathbf{L}_{33} & \mathbf{L}_{33} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{x}}_{0:k-1} \\ \hat{\mathbf{x}}_k \\ \hat{\mathbf{x}}_{k+1:K} \end{bmatrix} = \begin{bmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \\ \mathbf{r}_3 \end{bmatrix}, \quad (3.93)$$

where we have added the $(\hat{\cdot})$ to indicate this is the solution to the optimization estimation problem considered earlier. Our short-term goal, in making progress toward a recursive LG estimator, is to solve for $\hat{\mathbf{x}}_k$. To isolate $\hat{\mathbf{x}}_k$, we left-multiply both sides of (3.93) by

$$\begin{bmatrix} \mathbf{1} \\ -\mathbf{L}_{12}^T \mathbf{L}_{11}^{-1} & \mathbf{1} & -\mathbf{L}_{32}^T \mathbf{L}_{33}^{-1} \\ & \mathbf{1} & \end{bmatrix}, \quad (3.94)$$

which can be viewed as performing an elementary row operation (and therefore will not change the solution to (3.93)). The resulting system

of equations is

$$\begin{bmatrix} \mathbf{L}_{11} & \mathbf{L}_{12} \\ \mathbf{L}_{22} - \mathbf{L}_{12}^T \mathbf{L}_{11}^{-1} \mathbf{L}_{12} - \mathbf{L}_{32}^T \mathbf{L}_{33}^{-1} \mathbf{L}_{32} & \mathbf{L}_{32} \\ & \mathbf{L}_{33} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{x}}_{0:k-1} \\ \hat{\mathbf{x}}_k \\ \hat{\mathbf{x}}_{k+1:K} \end{bmatrix} = \begin{bmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 - \mathbf{L}_{12}^T \mathbf{L}_{11}^{-1} \mathbf{r}_1 - \mathbf{L}_{32}^T \mathbf{L}_{33}^{-1} \mathbf{r}_3 \\ \mathbf{r}_3 \end{bmatrix}, \quad (3.95)$$

and the solution for $\hat{\mathbf{x}}_k$ is therefore given by

$$\underbrace{(\mathbf{L}_{22} - \mathbf{L}_{12}^T \mathbf{L}_{11}^{-1} \mathbf{L}_{12} - \mathbf{L}_{32}^T \mathbf{L}_{33}^{-1} \mathbf{L}_{32})}_{\hat{\mathbf{P}}_k^{-1}} \hat{\mathbf{x}}_k = \underbrace{(\mathbf{r}_2 - \mathbf{L}_{12}^T \mathbf{L}_{11}^{-1} \mathbf{r}_1 - \mathbf{L}_{32}^T \mathbf{L}_{33}^{-1} \mathbf{r}_3)}_{\mathbf{q}_k}, \quad (3.96)$$

where we have defined $\hat{\mathbf{P}}_k$ (by its inverse) as well as \mathbf{q}_k . We have essentially marginalized out $\hat{\mathbf{x}}_{0:k-1}$ and $\hat{\mathbf{x}}_{k+1:K}$ just as in (3.85). We can now substitute the values of the \mathbf{L}_{ij} blocks back into $\hat{\mathbf{P}}_k^{-1}$ to see that

$$\begin{aligned} \hat{\mathbf{P}}_k^{-1} &= \mathbf{L}_{22} - \mathbf{L}_{12}^T \mathbf{L}_{11}^{-1} \mathbf{L}_{12} - \mathbf{L}_{32}^T \mathbf{L}_{33}^{-1} \mathbf{L}_{32} \\ &= \underbrace{\mathbf{H}_{22}^T \left(\mathbf{W}_2^{-1} - \mathbf{W}_2^{-1} \mathbf{H}_{21} \left(\mathbf{H}_{11}^T \mathbf{W}_1^{-1} \mathbf{H}_{11} + \mathbf{H}_{21}^T \mathbf{W}_2^{-1} \mathbf{H}_{21} \right)^{-1} \mathbf{H}_{21}^T \mathbf{W}_2^{-1} \right) \mathbf{H}_{22}}_{\hat{\mathbf{P}}_{k,f}^{-1} = \mathbf{H}_{22}^T \left(\mathbf{W}_2 + \mathbf{H}_{21} \left(\mathbf{H}_{11}^T \mathbf{W}_1^{-1} \mathbf{H}_{11} \right)^{-1} \mathbf{H}_{21}^T \right)^{-1} \mathbf{H}_{22}, \text{ by (2.75)}} \\ &\quad + \underbrace{\mathbf{H}_{32}^T \left(\mathbf{W}_3^{-1} - \mathbf{W}_3^{-1} \mathbf{H}_{33} \left(\mathbf{H}_{33}^T \mathbf{W}_3^{-1} \mathbf{H}_{33} + \mathbf{H}_{43}^T \mathbf{W}_4^{-1} \mathbf{H}_{43} \right)^{-1} \mathbf{H}_{33}^T \mathbf{W}_3^{-1} \right) \mathbf{H}_{32}}_{\hat{\mathbf{P}}_{k,b}^{-1} = \mathbf{H}_{32}^T \left(\mathbf{W}_3 + \mathbf{H}_{33} \left(\mathbf{H}_{43}^T \mathbf{W}_4^{-1} \mathbf{H}_{43} \right)^{-1} \mathbf{H}_{33}^T \right)^{-1} \mathbf{H}_{32}, \text{ by (2.75)}} \\ &= \underbrace{\hat{\mathbf{P}}_{k,f}^{-1}}_{\text{forward}} + \underbrace{\hat{\mathbf{P}}_{k,b}^{-1}}_{\text{backward}}, \end{aligned} \quad (3.97)$$

where the term labelled ‘forward’ depends only on the blocks of \mathbf{H} and \mathbf{W} up to time k and the term labelled ‘backward’ depends only on the blocks of \mathbf{H} and \mathbf{W} from $k+1$ to K . Turning now to \mathbf{q}_k , we substitute in the values of the \mathbf{L}_{ij} and \mathbf{r}_i blocks:

$$\begin{aligned} \mathbf{q}_k &= \mathbf{r}_2 - \mathbf{L}_{12}^T \mathbf{L}_{11}^{-1} \mathbf{r}_1 - \mathbf{L}_{32}^T \mathbf{L}_{33}^{-1} \mathbf{r}_3 \\ &= \underbrace{\mathbf{q}_{k,f}}_{\text{forward}} + \underbrace{\mathbf{q}_{k,b}}_{\text{backward}}, \end{aligned} \quad (3.98)$$

where again the term labelled ‘forward’ depends only on quantities up to time k and the term labelled ‘backward’ depends only on quantities

from time $k + 1$ to K . We made use of the following definitions:

$$\begin{aligned} \mathbf{q}_{k,f} &= -\mathbf{H}_{22}^T \mathbf{W}_2^{-1} \mathbf{H}_{21} (\mathbf{H}_{11}^T \mathbf{W}_1^{-1} \mathbf{H}_{11} + \mathbf{H}_{21}^T \mathbf{W}_2^{-1} \mathbf{H}_{21})^{-1} \mathbf{H}_{11}^T \mathbf{W}_1^{-1} \mathbf{z}_1 \\ &\quad + \mathbf{H}_{22}^T (\mathbf{W}_2^{-1} - \mathbf{W}_2^{-1} \mathbf{H}_{21} (\mathbf{H}_{11}^T \mathbf{W}_1^{-1} \mathbf{H}_{11} + \mathbf{H}_{21}^T \mathbf{W}_2^{-1} \mathbf{H}_{21})^{-1} \mathbf{H}_{21}^T \mathbf{W}_2^{-1}) \mathbf{z}_2, \end{aligned} \quad (3.99a)$$

$$\begin{aligned} \mathbf{q}_{k,b} &= \mathbf{H}_{32}^T (\mathbf{W}_3^{-1} - \mathbf{W}_3^{-1} \mathbf{H}_{33} (\mathbf{H}_{33}^T \mathbf{W}_3^{-1} \mathbf{H}_{33} + \mathbf{H}_{43}^T \mathbf{W}_4^{-1} \mathbf{H}_{43})^{-1} \mathbf{H}_{33}^T \mathbf{W}_3^{-1}) \mathbf{z}_3 \\ &\quad - \mathbf{H}_{32}^T \mathbf{W}_3^{-1} \mathbf{H}_{33} (\mathbf{H}_{43}^T \mathbf{W}_4^{-1} \mathbf{H}_{43} + \mathbf{H}_{33}^T \mathbf{W}_3^{-1} \mathbf{H}_{33})^{-1} \mathbf{H}_{43}^T \mathbf{W}_4^{-1} \mathbf{z}_4. \end{aligned} \quad (3.99b)$$

Now let us define the following two ‘forward’ and ‘backward’ estimators, $\hat{\mathbf{x}}_{k,f}$ and $\hat{\mathbf{x}}_{k,b}$, respectively:

$$\hat{\mathbf{P}}_{k,f}^{-1} \hat{\mathbf{x}}_{k,f} = \mathbf{q}_{k,f}, \quad (3.100a)$$

$$\hat{\mathbf{P}}_{k,b}^{-1} \hat{\mathbf{x}}_{k,b} = \mathbf{q}_{k,b}, \quad (3.100b)$$

where $\hat{\mathbf{x}}_{k,f}$ depends only on quantities up to time k and $\hat{\mathbf{x}}_{k,b}$ depends only on quantities from time $k + 1$ to K . Under these definitions we have that

$$\hat{\mathbf{P}}_k^{-1} = \hat{\mathbf{P}}_{k,f}^{-1} + \hat{\mathbf{P}}_{k,b}^{-1}, \quad (3.101)$$

$$\hat{\mathbf{P}}_k^{-1} \hat{\mathbf{x}}_k = \hat{\mathbf{P}}_{k,f}^{-1} \hat{\mathbf{x}}_{k,f} + \hat{\mathbf{P}}_{k,b}^{-1} \hat{\mathbf{x}}_{k,b}, \quad (3.102)$$

which is precisely the normalized product of two Gaussian PDFs, as was discussed in Section 2.2.6. Referring back to (3.86), we have that

$$p(\mathbf{x}_k | \mathbf{v}, \mathbf{y}) \rightarrow \mathcal{N}(\hat{\mathbf{x}}_k, \hat{\mathbf{P}}_k), \quad (3.103a)$$

$$p(\mathbf{x}_k | \hat{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k}) \rightarrow \mathcal{N}(\hat{\mathbf{x}}_{k,f}, \hat{\mathbf{P}}_{k,f}), \quad (3.103b)$$

$$p(\mathbf{x}_k | \mathbf{v}_{k+1:K}, \mathbf{y}_{k+1:K}) \rightarrow \mathcal{N}(\hat{\mathbf{x}}_{k,b}, \hat{\mathbf{P}}_{k,b}). \quad (3.103c)$$

where $\hat{\mathbf{P}}_k$, $\hat{\mathbf{P}}_{k,f}$, and $\hat{\mathbf{P}}_{k,b}$ are the covariances associated with $\hat{\mathbf{x}}_k$, $\hat{\mathbf{x}}_{k,f}$, and $\hat{\mathbf{x}}_{k,b}$. In other words we have Gaussian estimators with the MAP estimators as the means.

In the next section, we will examine how we can turn the forward Gaussian estimator, $\hat{\mathbf{x}}_{k,f}$, into a recursive filter¹⁶.

3.3.2 Kalman Filter via MAP

In this section, we will show how to turn the forward estimator from the last section into a recursive filter called the *Kalman filter* (Kalman, 1960b) using our MAP approach. To simplify the notation slightly, we will use $\hat{\mathbf{x}}_k$ instead of $\hat{\mathbf{x}}_{k,f}$ and $\hat{\mathbf{P}}_k$ instead of $\hat{\mathbf{P}}_{k,f}$, but these new symbols should not be confused with the batch/smoothed estimates discussed

¹⁶ A similar thing can be done for the backwards estimator, but the recursion is backwards in time rather than forwards.

previously. Let us assume we already have a forwards estimate and the associated covariance at some time $k - 1$:

$$\{\hat{\mathbf{x}}_{k-1}, \hat{\mathbf{P}}_{k-1}\}. \quad (3.104)$$

Recall that these estimates are based on all the data up to and including those at time $k - 1$. Our goal will be to compute

$$\{\hat{\mathbf{x}}_k, \hat{\mathbf{P}}_k\}, \quad (3.105)$$

using all the data up to and including those at time k . It turns out we do not need to start all over again, but rather can simply incorporate the new data at time k , \mathbf{v}_k and \mathbf{y}_k , into the estimate at time $k - 1$:

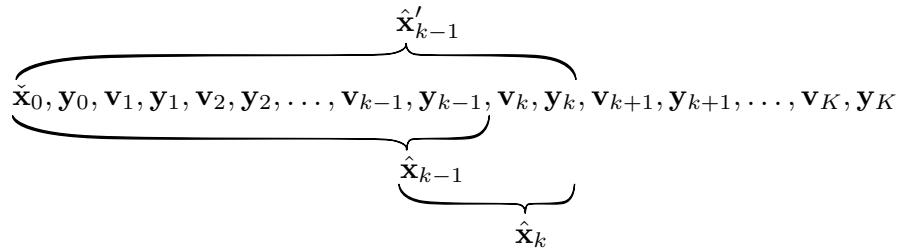
$$\{\hat{\mathbf{x}}_{k-1}, \hat{\mathbf{P}}_{k-1}, \mathbf{v}_k, \mathbf{y}_k\} \mapsto \{\hat{\mathbf{x}}_k, \hat{\mathbf{P}}_k\}. \quad (3.106)$$

To see this, we define

$$\mathbf{z} = \begin{bmatrix} \hat{\mathbf{x}}_{k-1} \\ \mathbf{v}_k \\ \mathbf{y}_k \end{bmatrix}, \quad \mathbf{H} = \begin{bmatrix} \mathbf{1} \\ -\mathbf{A}_{k-1} & \mathbf{1} \\ \mathbf{C}_k \end{bmatrix}, \quad \mathbf{W} = \begin{bmatrix} \hat{\mathbf{P}}_{k-1} & \mathbf{Q}_k & \mathbf{R}_k \end{bmatrix}, \quad (3.107)$$

where $\{\hat{\mathbf{x}}_{k-1}, \hat{\mathbf{P}}_{k-1}\}$ serve as substitutes for all the data up to time $k - 1$ ¹⁷. Figure 3.4 depicts this graphically.

Figure 3.4
Recursive filter
replaces past data
with an estimate.



Our usual MAP solution to the problem is $\hat{\mathbf{x}}$ given by

$$(\mathbf{H}^T \mathbf{W}^{-1} \mathbf{H}) \hat{\mathbf{x}} = \mathbf{H}^T \mathbf{W}^{-1} \mathbf{z}. \quad (3.108)$$

We then define

$$\hat{\mathbf{x}} = \begin{bmatrix} \hat{\mathbf{x}}'_{k-1} \\ \hat{\mathbf{x}}_k \end{bmatrix}, \quad (3.109)$$

where we carefully distinguish $\hat{\mathbf{x}}'_{k-1}$ from $\hat{\mathbf{x}}_{k-1}$. The addition of the ' indicates that $\hat{\mathbf{x}}'_{k-1}$ is the estimate at time $k - 1$ incorporating data up to and including time k , whereas $\hat{\mathbf{x}}_{k-1}$ is the estimate at time $k - 1$

¹⁷ To do this, we have actually employed something called the *Markov property*. Further discussion of this will be left to the chapter on nonlinear-non-Gaussian estimation techniques. For now it suffices to say that for LG estimation, this assumption is valid.

using data up to and including time $k-1$. Substituting in our quantities from (3.107) to the least-squares solution, we have

$$\begin{bmatrix} \hat{\mathbf{P}}_{k-1}^{-1} + \mathbf{A}_{k-1}^T \mathbf{Q}_k^{-1} \mathbf{A}_{k-1} & -\mathbf{A}_{k-1}^T \mathbf{Q}_k^{-1} \\ -\mathbf{Q}_k^{-1} \mathbf{A}_{k-1} & \mathbf{Q}_k^{-1} + \mathbf{C}_k^T \mathbf{R}_k^{-1} \mathbf{C}_k \end{bmatrix} \begin{bmatrix} \hat{\mathbf{x}}'_{k-1} \\ \hat{\mathbf{x}}_k \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{P}}_{k-1}^{-1} \hat{\mathbf{x}}_{k-1} - \mathbf{A}_{k-1}^T \mathbf{Q}_k^{-1} \mathbf{v}_k \\ \mathbf{Q}_k^{-1} \mathbf{v}_k + \mathbf{C}_k^T \mathbf{R}_k^{-1} \mathbf{y}_k \end{bmatrix}. \quad (3.110)$$

We do not really care what $\hat{\mathbf{x}}'_{k-1}$ is in this context, because we seek a recursive estimator appropriate to online estimation, and this quantity incorporates future data; we can marginalize this out by left-multiplying both sides by

$$\begin{bmatrix} \mathbf{1} & \mathbf{0} \\ \mathbf{Q}_k^{-1} \mathbf{A}_{k-1} \left(\hat{\mathbf{P}}_{k-1}^{-1} + \mathbf{A}_{k-1}^T \mathbf{Q}_k^{-1} \mathbf{A}_{k-1} \right)^{-1} & \mathbf{1} \end{bmatrix}, \quad (3.111)$$

which is just an elementary row operation and will not alter the solution to the linear system of equations¹⁸. Equation (3.110) then becomes

$$\begin{bmatrix} \hat{\mathbf{P}}_{k-1}^{-1} + \mathbf{A}_{k-1}^T \mathbf{Q}_k^{-1} \mathbf{A}_{k-1} & -\mathbf{A}_{k-1}^T \mathbf{Q}_k^{-1} \\ \mathbf{0} & \mathbf{Q}_k^{-1} - \mathbf{Q}_k^{-1} \mathbf{A}_{k-1} \left(\hat{\mathbf{P}}_{k-1}^{-1} + \mathbf{A}_{k-1}^T \mathbf{Q}_k^{-1} \mathbf{A}_{k-1} \right)^{-1} \times \mathbf{A}_{k-1}^T \mathbf{Q}_k^{-1} + \mathbf{C}_k^T \mathbf{R}_k^{-1} \mathbf{C}_k \end{bmatrix} \begin{bmatrix} \hat{\mathbf{x}}'_{k-1} \\ \hat{\mathbf{x}}_k \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{P}}_{k-1}^{-1} \hat{\mathbf{x}}_{k-1} - \mathbf{A}_{k-1}^T \mathbf{Q}_k^{-1} \mathbf{v}_k \\ \mathbf{Q}_k^{-1} \mathbf{A}_{k-1} \left(\hat{\mathbf{P}}_{k-1}^{-1} + \mathbf{A}_{k-1}^T \mathbf{Q}_k^{-1} \mathbf{A}_{k-1} \right)^{-1} \left(\hat{\mathbf{P}}_{k-1}^{-1} \hat{\mathbf{x}}_{k-1} - \mathbf{A}_{k-1}^T \mathbf{Q}_k^{-1} \mathbf{v}_k \right) + \mathbf{Q}_k^{-1} \mathbf{v}_k + \mathbf{C}_k^T \mathbf{R}_k^{-1} \mathbf{y}_k \end{bmatrix}. \quad (3.112)$$

The solution for $\hat{\mathbf{x}}_k$ is given by

$$\begin{aligned} & \underbrace{\left(\mathbf{Q}_k^{-1} - \mathbf{Q}_k^{-1} \mathbf{A}_{k-1} \left(\hat{\mathbf{P}}_{k-1}^{-1} + \mathbf{A}_{k-1}^T \mathbf{Q}_k^{-1} \mathbf{A}_{k-1} \right)^{-1} \mathbf{A}_{k-1}^T \mathbf{Q}_k^{-1} \right)}_{(\mathbf{Q}_k + \mathbf{A}_{k-1} \hat{\mathbf{P}}_{k-1} \mathbf{A}_{k-1}^T)^{-1} \text{ by (2.75)}} \\ & \quad + \mathbf{C}_k^T \mathbf{R}_k^{-1} \mathbf{C}_k \Big) \hat{\mathbf{x}}_k \\ &= \left(\mathbf{Q}_k^{-1} \mathbf{A}_{k-1} \left(\hat{\mathbf{P}}_{k-1}^{-1} + \mathbf{A}_{k-1}^T \mathbf{Q}_k^{-1} \mathbf{A}_{k-1} \right)^{-1} \right. \\ & \quad \times \left. \left(\hat{\mathbf{P}}_{k-1}^{-1} \hat{\mathbf{x}}_{k-1} - \mathbf{A}_{k-1}^T \mathbf{Q}_k^{-1} \mathbf{v}_k \right) \right. \\ & \quad \left. + \mathbf{Q}_k^{-1} \mathbf{v}_k + \mathbf{C}_k^T \mathbf{R}_k^{-1} \mathbf{y}_k \right). \quad (3.113) \end{aligned}$$

¹⁸ This is also sometimes called the *Schur complement*.

We then define the following helpful quantities:

$$\check{\mathbf{P}}_k = \mathbf{Q}_k + \mathbf{A}_{k-1} \hat{\mathbf{P}}_{k-1} \mathbf{A}_{k-1}^T, \quad (3.114a)$$

$$\hat{\mathbf{P}}_k = (\check{\mathbf{P}}_k^{-1} + \mathbf{C}_k^T \mathbf{R}_k^{-1} \mathbf{C}_k)^{-1}. \quad (3.114b)$$

Equation (3.113) then becomes

$$\begin{aligned} \hat{\mathbf{P}}_k^{-1} \hat{\mathbf{x}}_k &= \mathbf{Q}_k^{-1} \mathbf{A}_{k-1} \left(\hat{\mathbf{P}}_{k-1}^{-1} + \mathbf{A}_{k-1}^T \mathbf{Q}_k^{-1} \mathbf{A}_{k-1} \right)^{-1} \\ &\quad \times \left(\hat{\mathbf{P}}_{k-1}^{-1} \hat{\mathbf{x}}_{k-1} - \mathbf{A}_{k-1}^T \mathbf{Q}_k^{-1} \mathbf{v}_k \right) + \mathbf{Q}_k^{-1} \mathbf{v}_k + \mathbf{C}_k^T \mathbf{R}_k^{-1} \mathbf{y}_k \\ &= \underbrace{\mathbf{Q}_k^{-1} \mathbf{A}_{k-1} \left(\hat{\mathbf{P}}_{k-1}^{-1} + \mathbf{A}_{k-1}^T \mathbf{Q}_k^{-1} \mathbf{A}_{k-1} \right)^{-1} \hat{\mathbf{P}}_{k-1}^{-1} \hat{\mathbf{x}}_{k-1}}_{\check{\mathbf{P}}_k^{-1} \mathbf{A}_{k-1} \text{ by logic below}} \\ &\quad + \underbrace{\left(\mathbf{Q}_k^{-1} - \mathbf{Q}_k^{-1} \mathbf{A}_{k-1} \left(\hat{\mathbf{P}}_{k-1}^{-1} + \mathbf{A}_{k-1}^T \mathbf{Q}_k^{-1} \mathbf{A}_{k-1} \right)^{-1} \mathbf{A}_{k-1}^T \mathbf{Q}_k^{-1} \right) \mathbf{v}_k}_{\check{\mathbf{P}}_k^{-1}} \\ &\quad + \mathbf{C}_k^T \mathbf{R}_k^{-1} \mathbf{y}_k \\ &= \check{\mathbf{P}}_k^{-1} \underbrace{(\mathbf{A}_{k-1} \hat{\mathbf{x}}_{k-1} + \mathbf{v}_k)}_{\hat{\mathbf{x}}_k} + \mathbf{C}_k^T \mathbf{R}_k^{-1} \mathbf{y}_k, \end{aligned} \quad (3.115)$$

where we have defined $\hat{\mathbf{x}}_k$ as the ‘predicted’ value of the state. We also made use of the following logic in simplifying the above:

$$\begin{aligned} &\mathbf{Q}_k^{-1} \mathbf{A}_{k-1} \underbrace{\left(\hat{\mathbf{P}}_{k-1}^{-1} + \mathbf{A}_{k-1}^T \mathbf{Q}_k^{-1} \mathbf{A}_{k-1} \right)^{-1} \hat{\mathbf{P}}_{k-1}^{-1}}_{\text{apply (2.75) again}} \\ &= \mathbf{Q}_k^{-1} \mathbf{A}_{k-1} \left(\hat{\mathbf{P}}_{k-1} - \hat{\mathbf{P}}_{k-1} \mathbf{A}_{k-1}^T \underbrace{\left(\mathbf{Q}_k + \mathbf{A}_{k-1} \hat{\mathbf{P}}_{k-1} \mathbf{A}_{k-1}^T \right)^{-1}}_{\check{\mathbf{P}}_k^{-1}} \right. \\ &\quad \times \left. \mathbf{A}_{k-1} \hat{\mathbf{P}}_{k-1} \right) \hat{\mathbf{P}}_{k-1}^{-1} \\ &= \left(\mathbf{Q}_k^{-1} - \mathbf{Q}_k^{-1} \underbrace{\mathbf{A}_{k-1} \hat{\mathbf{P}}_{k-1} \mathbf{A}_{k-1}^T}_{\check{\mathbf{P}}_k - \mathbf{Q}_k} \check{\mathbf{P}}_k^{-1} \right) \mathbf{A}_{k-1} \\ &= \left(\mathbf{Q}_k^{-1} - \mathbf{Q}_k^{-1} + \check{\mathbf{P}}_k^{-1} \right) \mathbf{A}_{k-1} \\ &= \check{\mathbf{P}}_k^{-1} \mathbf{A}_{k-1}. \end{aligned} \quad (3.116)$$

Bringing together all of the above, we have for the recursive filter update the following:

$$\text{predictor: } \check{\mathbf{P}}_k = \mathbf{A}_{k-1} \hat{\mathbf{P}}_{k-1} \mathbf{A}_{k-1}^T + \mathbf{Q}_k, \quad (3.117a)$$

$$\check{\mathbf{x}}_k = \mathbf{A}_{k-1} \hat{\mathbf{x}}_{k-1} + \mathbf{v}_k, \quad (3.117b)$$

$$\text{corrector: } \hat{\mathbf{P}}_k^{-1} = \check{\mathbf{P}}_k^{-1} + \mathbf{C}_k^T \mathbf{R}_k^{-1} \mathbf{C}_k, \quad (3.117c)$$

$$\hat{\mathbf{P}}_k^{-1} \hat{\mathbf{x}}_k = \check{\mathbf{P}}_k^{-1} \check{\mathbf{x}}_k + \mathbf{C}_k^T \mathbf{R}_k^{-1} \mathbf{y}_k, \quad (3.117d)$$

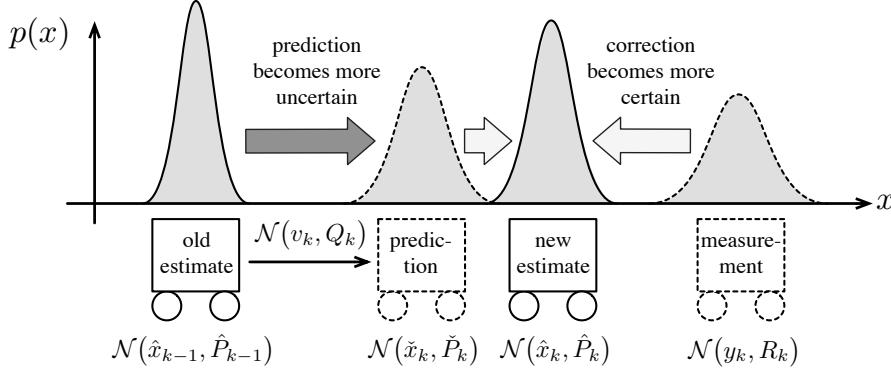


Figure 3.5 The Kalman filter works in two steps: prediction then correction. The prediction step propagates the old estimate, \hat{x}_{k-1} , forward in time using the measurement model and latest input, v_k , to arrive at the prediction, \check{x}_k . The correction step fuses the prediction with the latest measurement, y_k , to arrive at the new estimate, \hat{x}_k ; this step is carried out using a normalized product of Gaussians (clear from inverse covariance version of KF).

which we will refer to as *inverse covariance* or information form for the Kalman filter. Figure 3.5 depicts the predictor-corrector form of the Kalman filter graphically.

To get to *canonical form*, we manipulate these equations slightly. Begin by defining the *Kalman gain*, \mathbf{K}_k , as

$$\mathbf{K}_k = \hat{\mathbf{P}}_k \mathbf{C}_k^T \mathbf{R}_k^{-1}. \quad (3.118)$$

We then manipulate:

$$\begin{aligned} \mathbf{I} &= \hat{\mathbf{P}}_k (\check{\mathbf{P}}_k^{-1} + \mathbf{C}_k^T \mathbf{R}_k^{-1} \mathbf{C}_k) \\ &= \hat{\mathbf{P}}_k \check{\mathbf{P}}_k^{-1} + \mathbf{K}_k \mathbf{C}_k, \end{aligned} \quad (3.119a)$$

$$\hat{\mathbf{P}}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{C}_k) \check{\mathbf{P}}_k, \quad (3.119b)$$

$$\underbrace{\hat{\mathbf{P}}_k \mathbf{C}_k^T \mathbf{R}_k^{-1}}_{\mathbf{K}_k} = (\mathbf{I} - \mathbf{K}_k \mathbf{C}_k) \check{\mathbf{P}}_k \mathbf{C}_k^T \mathbf{R}_k^{-1}, \quad (3.119c)$$

$$\mathbf{K}_k (\mathbf{I} + \mathbf{C}_k \check{\mathbf{P}}_k \mathbf{C}_k^T \mathbf{R}_k^{-1}) = \check{\mathbf{P}}_k \mathbf{C}_k^T \mathbf{R}_k^{-1}. \quad (3.119d)$$

Solving for \mathbf{K}_k in this last expression, we can rewrite the recursive filter equations as

$$\text{predictor: } \check{\mathbf{P}}_k = \mathbf{A}_{k-1} \hat{\mathbf{P}}_{k-1} \mathbf{A}_{k-1}^T + \mathbf{Q}_k, \quad (3.120a)$$

$$\check{\mathbf{x}}_k = \mathbf{A}_{k-1} \hat{\mathbf{x}}_{k-1} + \mathbf{v}_k, \quad (3.120b)$$

$$\text{Kalman gain: } \mathbf{K}_k = \check{\mathbf{P}}_k \mathbf{C}_k^T (\mathbf{C}_k \check{\mathbf{P}}_k \mathbf{C}_k^T + \mathbf{R}_k)^{-1}, \quad (3.120c)$$

$$\text{corrector: } \hat{\mathbf{P}}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{C}_k) \check{\mathbf{P}}_k, \quad (3.120d)$$

$$\hat{\mathbf{x}}_k = \check{\mathbf{x}}_k + \mathbf{K}_k \underbrace{(\mathbf{y}_k - \mathbf{C}_k \check{\mathbf{x}}_k)}_{\text{innovation}}, \quad (3.120e)$$

where the *innovation* has been highlighted; it is the difference between the actual and expected measurements. The role of the Kalman gain is to properly weight the innovation's contribution to the estimate (in comparison to the prediction). In this form, these five equations (and their extension to nonlinear systems) have been the workhorse of estimation since Kalman's initial paper (Kalman, 1960b). These are identi-

cal to the forward pass of the Rauch-Tung-Striebel smoother discussed previously (with the $(\cdot)_f$ subscripts dropped).

3.3.3 Kalman Filter via Bayesian Inference

A cleaner, simpler derivation of the Kalman filter can be had using our Bayesian inference approach¹⁹. Our Gaussian prior estimate at $k - 1$ is

$$p(\mathbf{x}_{k-1} | \check{\mathbf{x}}_0, \mathbf{v}_{1:k-1}, \mathbf{y}_{0:k-1}) = \mathcal{N}(\hat{\mathbf{x}}_{k-1}, \hat{\mathbf{P}}_{k-1}). \quad (3.121)$$

First, for the *prediction step*, we incorporate the latest input, \mathbf{v}_k , to write a ‘prior’ at time k :

$$p(\mathbf{x}_k | \check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k-1}) = \mathcal{N}(\check{\mathbf{x}}_k, \check{\mathbf{P}}_k), \quad (3.122)$$

where

$$\check{\mathbf{P}}_k = \mathbf{A}_{k-1} \hat{\mathbf{P}}_{k-1} \mathbf{A}_{k-1}^T + \mathbf{Q}_k, \quad (3.123a)$$

$$\check{\mathbf{x}}_k = \mathbf{A}_{k-1} \hat{\mathbf{x}}_{k-1} + \mathbf{v}_k. \quad (3.123b)$$

These are identical to the prediction equations from the previous section. These last two expressions can be found by exactly passing the prior at $k - 1$ through the linear motion model. For the mean we have

$$\begin{aligned} \check{\mathbf{x}}_k &= E[\mathbf{x}_k] = E[\mathbf{A}_{k-1} \mathbf{x}_{k-1} + \mathbf{v}_k + \mathbf{w}_k] \\ &= \mathbf{A}_{k-1} \underbrace{E[\mathbf{x}_{k-1}]}_{\hat{\mathbf{x}}_{k-1}} + \mathbf{v}_k + \underbrace{E[\mathbf{w}_k]}_0 = \mathbf{A}_{k-1} \hat{\mathbf{x}}_{k-1} + \mathbf{v}_k, \end{aligned} \quad (3.124)$$

and for the covariance we have

$$\begin{aligned} \check{\mathbf{P}}_k &= E[(\mathbf{x}_k - E[\mathbf{x}_k])(\mathbf{x}_k - E[\mathbf{x}_k])^T] \\ &= E[(\mathbf{A}_{k-1} \mathbf{x}_{k-1} + \mathbf{v}_k + \mathbf{w}_k - \mathbf{A}_{k-1} \hat{\mathbf{x}}_{k-1} - \mathbf{v}_k) \\ &\quad \times (\mathbf{A}_{k-1} \mathbf{x}_{k-1} + \mathbf{v}_k + \mathbf{w}_k - \mathbf{A}_{k-1} \hat{\mathbf{x}}_{k-1} - \mathbf{v}_k)^T] \\ &= \mathbf{A}_{k-1} \underbrace{E[(\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1})(\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1})^T]}_{\hat{\mathbf{P}}_{k-1}} \mathbf{A}_{k-1}^T + \underbrace{E[\mathbf{w}_k \mathbf{w}_k^T]}_{\mathbf{Q}_k} \\ &= \mathbf{A}_{k-1} \hat{\mathbf{P}}_{k-1} \mathbf{A}_{k-1}^T + \mathbf{Q}_k. \end{aligned} \quad (3.125)$$

Next, for the *correction step*, we express the joint density of our state

¹⁹ In the next chapter, we will generalize this section to present the *Bayes filter*, which can handle non-Gaussian PDFs as well as nonlinear motion and observation models. We can think of this section as a special case of the Bayes filter, one that requires no approximations to be made.

and latest measurement, at time k , as a Gaussian:

$$\begin{aligned} p(\mathbf{x}_k, \mathbf{y}_k | \check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k-1}) &= \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_x \\ \boldsymbol{\mu}_y \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{xx} & \boldsymbol{\Sigma}_{xy} \\ \boldsymbol{\Sigma}_{yx} & \boldsymbol{\Sigma}_{yy} \end{bmatrix}\right) \\ &= \mathcal{N}\left(\begin{bmatrix} \check{\mathbf{x}}_k \\ \mathbf{C}_k \check{\mathbf{x}}_k \end{bmatrix}, \begin{bmatrix} \check{\mathbf{P}}_k & \check{\mathbf{P}}_k \mathbf{C}_k^T \\ \mathbf{C}_k \check{\mathbf{P}}_k & \mathbf{C}_k \check{\mathbf{P}}_k \mathbf{C}_k^T + \mathbf{R}_k \end{bmatrix}\right). \end{aligned} \quad (3.126)$$

Looking back to Section 2.2.3, where we introduced Bayesian inference, we can then directly write the conditional density for \mathbf{x}_k (i.e., the posterior) as

$$\begin{aligned} p(\mathbf{x}_k | \check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k}) &= \mathcal{N}\left(\underbrace{\boldsymbol{\mu}_x + \boldsymbol{\Sigma}_{xy} \boldsymbol{\Sigma}_{yy}^{-1} (\mathbf{y}_k - \boldsymbol{\mu}_y)}_{\hat{\mathbf{x}}_k}, \underbrace{\boldsymbol{\Sigma}_{xx} - \boldsymbol{\Sigma}_{xy} \boldsymbol{\Sigma}_{yy}^{-1} \boldsymbol{\Sigma}_{yx}}_{\hat{\mathbf{P}}_k}\right), \end{aligned} \quad (3.127)$$

where we have defined $\hat{\mathbf{x}}_k$ as the mean and $\hat{\mathbf{P}}_k$ as the covariance. Substituting in the moments from above, we have

$$\mathbf{K}_k = \check{\mathbf{P}}_k \mathbf{C}_k^T (\mathbf{C}_k \check{\mathbf{P}}_k \mathbf{C}_k^T + \mathbf{R}_k)^{-1}, \quad (3.128a)$$

$$\hat{\mathbf{P}}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{C}_k) \check{\mathbf{P}}_k, \quad (3.128b)$$

$$\hat{\mathbf{x}}_k = \check{\mathbf{x}}_k + \mathbf{K}_k (\mathbf{y}_k - \mathbf{C}_k \check{\mathbf{x}}_k), \quad (3.128c)$$

which are identical to the correction equations from the previous section on MAP. Again, this is because the motion and measurement models are *linear* and the noises and prior are Gaussian. Under these conditions, the posterior density is exactly Gaussian. Thus, the *mean* and *mode* of the posterior are one and the same. This property does not hold if we switch to a *nonlinear* measurement model, which we discuss in the next chapter.

3.3.4 Kalman Filter via Gain Optimization

The Kalman filter is often referred to as being *optimal*. We did indeed perform an optimization to come up with the recursive relations above in the MAP derivation. There are also several other ways to look at the optimality of the KF. We present one of these.

Assume we have an estimator with the correction step taking the form

$$\hat{\mathbf{x}}_k = \check{\mathbf{x}}_k + \mathbf{K}_k (\mathbf{y}_k - \mathbf{C}_k \check{\mathbf{x}}_k), \quad (3.129)$$

but we do not yet know the gain matrix, \mathbf{K}_k , to blend the corrective measurements with the prediction. If we define the error in the state estimate to be

$$\hat{\mathbf{e}}_k = \hat{\mathbf{x}}_k - \mathbf{x}_k, \quad (3.130)$$

then we have²⁰

$$E[\hat{\mathbf{e}}_k \hat{\mathbf{e}}_k^T] = (\mathbf{1} - \mathbf{K}_k \mathbf{C}_k) \check{\mathbf{P}}_k (\mathbf{1} - \mathbf{K}_k \mathbf{C}_k)^T + \mathbf{K}_k \mathbf{R}_k \mathbf{K}_k^T. \quad (3.131)$$

We then define a cost function of the form

$$J(\mathbf{K}_k) = \frac{1}{2} \text{tr} E[\hat{\mathbf{e}}_k \hat{\mathbf{e}}_k^T] = E \left[\frac{1}{2} \hat{\mathbf{e}}_k^T \hat{\mathbf{e}}_k \right], \quad (3.132)$$

which quantifies (in some sense) the magnitude of the covariance of $\hat{\mathbf{e}}_k$. We can minimize this cost directly with respect to \mathbf{K}_k , to generate the ‘minimum variance’ estimate. We will make use of the identities

$$\frac{\partial \text{tr } \mathbf{XY}}{\partial \mathbf{X}} \equiv \mathbf{Y}^T, \quad \frac{\partial \text{tr } \mathbf{ZXZ}^T}{\partial \mathbf{X}} \equiv 2\mathbf{Z}, \quad (3.133)$$

where \mathbf{Z} is symmetric. Then we have

$$\frac{\partial J(\mathbf{K}_k)}{\partial \mathbf{K}_k} = -(\mathbf{1} - \mathbf{K}_k \mathbf{C}_k) \check{\mathbf{P}}_k \mathbf{C}_k^T + \mathbf{K}_k \mathbf{R}_k. \quad (3.134a)$$

Setting this to zero and solving for \mathbf{K}_k , we have

$$\mathbf{K}_k = \check{\mathbf{P}}_k \mathbf{C}_k^T (\mathbf{C}_k \check{\mathbf{P}}_k \mathbf{C}_k^T + \mathbf{R}_k)^{-1}, \quad (3.135)$$

which is our usual expression for the Kalman gain.

3.3.5 Kalman Filter Discussion

There are a few points worth mentioning:

- (i) For a linear system with Gaussian noise, the Kalman filter equations are the *best linear unbiased estimate (BLUE)*; this means they are performing right at the Cramér-Rao lower bound.
- (ii) Initial conditions must be provided, $\{\check{\mathbf{x}}_0, \check{\mathbf{P}}_0\}$.
- (iii) The covariance equations can be propagated independently of the mean equations. Sometimes a steady-state value of \mathbf{K}_k is computed and used for all time-steps to propagate the mean; this is known as the ‘steady-state Kalman filter’.
- (iv) At implementation, we must use $\mathbf{y}_{k,\text{meas}}$, the actual readings we receive from our sensors, in the filter.
- (v) A similar set of equations can be developed for the backwards estimator that runs backwards in time.

It is worth reminding ourselves that we have arrived at the Kalman filter equations through both an optimization paradigm and a full Bayesian paradigm. The difference between these two will be significant when we consider what happens in the nonlinear case (and why the extension of the Kalman filter, the *extended Kalman filter (EKF)*, does not perform well in many situations).

²⁰ This is sometimes referred to as the *Joseph form* of the covariance update.

3.3.6 Error Dynamics

It is useful to look at the difference between the estimated state and the actual state. We define the following errors:

$$\check{\mathbf{e}}_k = \check{\mathbf{x}}_k - \mathbf{x}_k, \quad (3.136a)$$

$$\hat{\mathbf{e}}_k = \hat{\mathbf{x}}_k - \mathbf{x}_k. \quad (3.136b)$$

Using (3.1) and (3.120), we can then write out the ‘error dynamics’:

$$\check{\mathbf{e}}_k = \mathbf{A}_{k-1} \hat{\mathbf{e}}_{k-1} - \mathbf{w}_k, \quad (3.137a)$$

$$\hat{\mathbf{e}}_k = (\mathbf{1} - \mathbf{K}_k \mathbf{C}_k) \check{\mathbf{e}}_k + \mathbf{K}_k \mathbf{n}_k, \quad (3.137b)$$

where we note that $\hat{\mathbf{e}}_0 = \hat{\mathbf{x}}_0 - \mathbf{x}_0$. From this system it is not hard to see that $E[\hat{\mathbf{e}}_k] = \mathbf{0}$ for $k > 0$ so long as $E[\hat{\mathbf{e}}_0] = \mathbf{0}$. This means our estimator is *unbiased*. We can use proof by induction. It is true for $k = 0$ by assertion. Assume it is also true for $k - 1$. Then

$$E[\check{\mathbf{e}}_k] = \mathbf{A}_{k-1} \underbrace{E[\hat{\mathbf{e}}_{k-1}]}_0 - \underbrace{E[\mathbf{w}_k]}_0 = \mathbf{0}, \quad (3.138a)$$

$$E[\hat{\mathbf{e}}_k] = (\mathbf{1} - \mathbf{K}_k \mathbf{C}_k) \underbrace{E[\check{\mathbf{e}}_k]}_0 + \mathbf{K}_k \underbrace{E[\mathbf{n}_k]}_0 = \mathbf{0}. \quad (3.138b)$$

It is therefore true for all k . It is less obvious that

$$E[\check{\mathbf{e}}_k \check{\mathbf{e}}_k^T] = \check{\mathbf{P}}_k, \quad (3.139a)$$

$$E[\hat{\mathbf{e}}_k \hat{\mathbf{e}}_k^T] = \hat{\mathbf{P}}_k, \quad (3.139b)$$

for $k > 0$ so long as $E[\hat{\mathbf{e}}_0 \hat{\mathbf{e}}_0^T] = \hat{\mathbf{P}}_0$. This means our estimator is *consistent*. We again use proof by induction. It is true for $k = 0$ by assertion. Assume $E[\hat{\mathbf{e}}_{k-1} \hat{\mathbf{e}}_{k-1}^T] = \hat{\mathbf{P}}_{k-1}$. Then

$$\begin{aligned} E[\check{\mathbf{e}}_k \check{\mathbf{e}}_k^T] &= E[(\mathbf{A}_{k-1} \hat{\mathbf{e}}_{k-1} - \mathbf{w}_k)(\mathbf{A}_{k-1} \hat{\mathbf{e}}_{k-1} - \mathbf{w}_k)^T] \\ &= \mathbf{A}_{k-1} \underbrace{E[\hat{\mathbf{e}}_{k-1} \hat{\mathbf{e}}_{k-1}^T]}_{\hat{\mathbf{P}}_{k-1}} \mathbf{A}_{k-1}^T - \mathbf{A}_{k-1} \underbrace{E[\hat{\mathbf{e}}_{k-1} \mathbf{w}_k^T]}_{\mathbf{0} \text{ by independence}} \\ &\quad - \underbrace{E[\mathbf{w}_k \hat{\mathbf{e}}_{k-1}^T]}_{\mathbf{0} \text{ by independence}} \mathbf{A}_{k-1}^T + \underbrace{E[\mathbf{w}_k \mathbf{w}_k^T]}_{\mathbf{Q}_k} \\ &= \check{\mathbf{P}}_k, \end{aligned} \quad (3.140)$$

and

$$\begin{aligned}
E[\hat{\mathbf{e}}_k \hat{\mathbf{e}}_k^T] &= E\left[\left((\mathbf{1} - \mathbf{K}_k \mathbf{C}_k) \check{\mathbf{e}}_k + \mathbf{K}_k \mathbf{n}_k\right) \left((\mathbf{1} - \mathbf{K}_k \mathbf{C}_k) \check{\mathbf{e}}_k + \mathbf{K}_k \mathbf{n}_k\right)^T\right] \\
&= (\mathbf{1} - \mathbf{K}_k \mathbf{C}_k) \underbrace{E[\check{\mathbf{e}}_k \check{\mathbf{e}}_k^T]}_{\check{\mathbf{P}}_k} (\mathbf{1} - \mathbf{K}_k \mathbf{C}_k)^T \\
&\quad + (\mathbf{1} - \mathbf{K}_k \mathbf{C}_k) \underbrace{E[\check{\mathbf{e}}_k \mathbf{n}_k^T]}_{\mathbf{0} \text{ by independence}} \mathbf{K}_k^T \\
&\quad + \mathbf{K}_k \underbrace{E[\mathbf{n}_k \check{\mathbf{e}}_k^T]}_{\mathbf{0} \text{ by independence}} (\mathbf{1} - \mathbf{K}_k \mathbf{C}_k)^T + \mathbf{K}_k \underbrace{E[\mathbf{n}_k \mathbf{n}_k^T]}_{\mathbf{R}_k} \mathbf{K}_k^T \\
&= (\mathbf{1} - \mathbf{K}_k \mathbf{C}_k) \check{\mathbf{P}}_k \underbrace{- \hat{\mathbf{P}}_k \mathbf{C}_k^T \mathbf{K}_k^T + \mathbf{K}_k \mathbf{R}_k \mathbf{K}_k^T}_{\mathbf{0} \text{ because } \mathbf{K}_k = \hat{\mathbf{P}}_k \mathbf{C}_k^T \mathbf{R}_k^{-1}} \\
&= \hat{\mathbf{P}}_k.
\end{aligned} \tag{3.141}$$

It is therefore true for all k . This means that the true uncertainty in the system (i.e., the covariance of the error, $E[\hat{\mathbf{e}}_k \hat{\mathbf{e}}_k^T]$) is perfectly modelled by our estimate of the covariance, $\hat{\mathbf{P}}_k$. In this sense, the Kalman filter is an optimal filter. This is why it is sometimes referred to as *best linear unbiased estimate (BLUE)*. Yet another way of saying this is that the covariance of the Kalman filter is right at the Cramér-Rao Lower Bound; we cannot be any more certain in our estimate given the uncertainty in the measurements we have used in that estimate.

A final important point to make is that the expectations we have employed in this section are over the possible outcomes of the random variables. They are not time averages. If were to run an infinite number of trials and average over the trials (i.e., an ensemble average), then we should see an average performance of zero error (i.e., an unbiased estimator). This does not imply that within a single trial (i.e., a realization) the error will be zero or decay to zero over time.

3.3.7 Existence, Uniqueness, and Observability

A sketch of the stability proof of the KF is provided (Simon, 2006). We consider only the time-invariant case and use italicized symbols to avoid confusion with the lifted form: $\mathbf{A}_k = \mathbf{A}$, $\mathbf{C}_k = \mathbf{C}$, $\mathbf{Q}_k = \mathbf{Q}$, $\mathbf{R}_k = \mathbf{R}$. The sketch proceeds as follows:

- (i) The covariance equation of the KF can be iterated to convergence prior to computing the equations for the mean. A big question is whether the covariance will converge to a steady-state value and, if so, whether it will be unique. Writing \mathbf{P} to mean the steady-state value for $\check{\mathbf{P}}_k$, we have (by combining the predictive and corrective covariance equations) that the follow-

ing must be true at steady state:

$$\mathbf{P} = \mathbf{A}(\mathbf{1} - \mathbf{K}\mathbf{C})\mathbf{P}(\mathbf{1} - \mathbf{K}\mathbf{C})^T\mathbf{A}^T + \mathbf{A}\mathbf{K}\mathbf{R}\mathbf{K}^T\mathbf{A}^T + \mathbf{Q}, \quad (3.142)$$

which is one form of the Discrete Algebraic Riccati Equation (DARE). Note that \mathbf{K} depends on \mathbf{P} in the above equation. The DARE has a unique positive-semidefinite solution, \mathbf{P} , if and only if the following conditions hold:

- $\mathbf{R} > 0$; note that we already assume this in the batch LG case,
- $\mathbf{Q} \geq 0$; in the batch LG case, we actually assumed that $\mathbf{Q} > 0$, whereupon the next condition is redundant,
- (\mathbf{A}, \mathbf{V}) is stabilizable with $\mathbf{V}^T\mathbf{V} = \mathbf{Q}$; this condition is redundant when $\mathbf{Q} > 0$,
- (\mathbf{A}, \mathbf{C}) is detectable; same as ‘observable’ except any unobservable eigenvalues are stable; we saw a similar observability condition in the batch LG case.

The proof of the above statement is beyond the scope of this book.

- (ii) Once the covariance evolves to its steady-state value, \mathbf{P} , so does the Kalman gain. Let \mathbf{K} be the steady-state value of \mathbf{K}_k . We have

$$\mathbf{K} = \mathbf{P}\mathbf{C}^T \left(\mathbf{C}\mathbf{P}\mathbf{C}^T + \mathbf{R} \right)^{-1} \quad (3.143)$$

for the steady-state Kalman gain.

- (iii) The error dynamics of the filter are then stable:

$$E[\check{\mathbf{e}}_k] = \underbrace{\mathbf{A}(\mathbf{1} - \mathbf{K}\mathbf{C})}_{\text{eigs. } < 1 \text{ in mag.}} E[\check{\mathbf{e}}_{k-1}]. \quad (3.144)$$

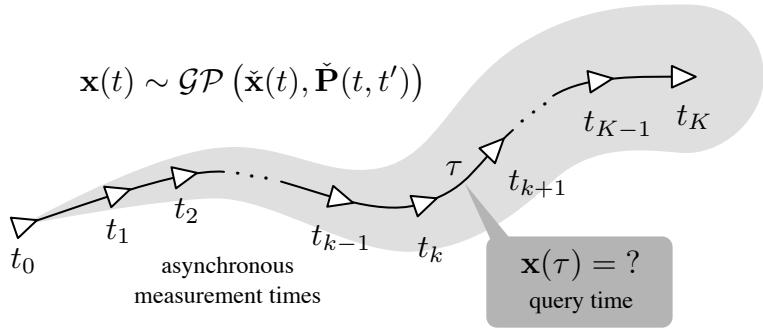
We can see this by noting that for any eigenvector, \mathbf{v} , corresponding to an eigenvalue, λ , of $(\mathbf{1} - \mathbf{K}\mathbf{C})^T\mathbf{A}^T$, we have

$$\begin{aligned} \mathbf{v}^T \mathbf{P} \mathbf{v} &= \underbrace{\mathbf{v}^T \mathbf{A}(\mathbf{1} - \mathbf{K}\mathbf{C}) \mathbf{P}}_{\lambda \mathbf{v}^T} \underbrace{(\mathbf{1} - \mathbf{K}\mathbf{C})^T \mathbf{A}^T \mathbf{v}}_{\lambda \mathbf{v}} \\ &\quad + \mathbf{v}^T (\mathbf{A}\mathbf{K}\mathbf{R}\mathbf{K}^T\mathbf{A}^T + \mathbf{Q}) \mathbf{v}, \end{aligned} \quad (3.145a)$$

$$(1 - \lambda^2) \underbrace{\mathbf{v}^T \mathbf{P} \mathbf{v}}_{>0} = \underbrace{\mathbf{v}^T (\mathbf{A}\mathbf{K}\mathbf{R}\mathbf{K}^T\mathbf{A}^T + \mathbf{Q}) \mathbf{v}}_{>0}, \quad (3.145b)$$

which means that we must have $|\lambda| < 1$, and thus the steady-state error dynamics are stable. Technically the right-hand side could be zero, but after N repetitions of this process, we build up a copy of the observability Grammian on the right-hand, side making it invertible (if the system is observable).

Figure 3.6 State estimation with a continuous-time prior can be viewed as a one-dimensional Gaussian process regression with time as the independent variable. We have data about the trajectory at a number of asynchronous measurement times and would like to query the state at some other time of interest.



3.4 Batch Continuous-Time Estimation

In this section, we circle back to consider a more general problem than the discrete-time setup in the earlier part of this chapter. In particular, we consider what happens when we choose to use a continuous-time motion model as the prior. We approach the problem from a *Gaussian process regression* perspective (Rasmussen and Williams, 2006). We show that for linear-Gaussian systems, the discrete-time formulation is implementing the continuous-time one exactly, under certain special conditions (Tong et al., 2013; Barfoot et al., 2014).

3.4.1 Gaussian Process Regression

We take a *Gaussian process regression* approach to state estimation²¹. This allows us (i) to represent trajectories in continuous time (and therefore query the solution at any time of interest) and, (ii) for the nonlinear case that we will treat in the next chapter, to optimize our solution by iterating over the entire trajectory (it is difficult to do this in the recursive formulation, which typically iterates at just one timestep at a time). We will show that under a certain special class of prior motion models, GP regression enjoys a sparse structure that allows for very efficient solutions.

We will consider systems with a continuous-time GP process model prior and a discrete-time, linear measurement model:

$$\mathbf{x}(t) \sim \mathcal{GP}(\check{\mathbf{x}}(t), \check{\mathbf{P}}(t, t')), \quad t_0 < t, t' \quad (3.146)$$

$$\mathbf{y}_k = \mathbf{C}_k \mathbf{x}(t_k) + \mathbf{n}_k, \quad t_0 < t_1 < \dots < t_K, \quad (3.147)$$

where $\mathbf{x}(t)$ is the state, $\check{\mathbf{x}}(t)$ is the mean function, $\check{\mathbf{P}}(t, t')$ is the covariance function, \mathbf{y}_k are measurements, $\mathbf{n}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k)$ is Gaussian measurement noise, and \mathbf{C}_k is the measurement model coefficient matrix.

We consider that we want to query the state at a number of times

²¹ There are other ways to represent continuous-time trajectories such as temporal basis functions; see Furgale et al. (2015) for a detailed review.

$(\tau_0 < \tau_1 < \dots < \tau_J)$ that may or may not be different from the measurement times $(t_0 < t_1 < \dots < t_K)$. Figure 3.6 depicts our problem setup. The joint density between the state (at the query times) and the measurements (at the measurement times) is written as

$$p\left(\begin{bmatrix} \mathbf{x}_\tau \\ \mathbf{y} \end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix} \check{\mathbf{x}}_\tau \\ \mathbf{C}\check{\mathbf{x}} \end{bmatrix}, \begin{bmatrix} \check{\mathbf{P}}_{\tau\tau} & \check{\mathbf{P}}_\tau \mathbf{C}^T \\ \mathbf{C}\check{\mathbf{P}}_\tau^T & \mathbf{R} + \mathbf{C}\check{\mathbf{P}}\mathbf{C}^T \end{bmatrix}\right), \quad (3.148)$$

where

$$\begin{aligned} \mathbf{x} &= \begin{bmatrix} \mathbf{x}(t_0) \\ \vdots \\ \mathbf{x}(t_K) \end{bmatrix}, \quad \check{\mathbf{x}} = \begin{bmatrix} \check{\mathbf{x}}(t_0) \\ \vdots \\ \check{\mathbf{x}}(t_K) \end{bmatrix}, \quad \mathbf{x}_\tau = \begin{bmatrix} \mathbf{x}(\tau_0) \\ \vdots \\ \mathbf{x}(\tau_J) \end{bmatrix}, \quad \check{\mathbf{x}}_\tau = \begin{bmatrix} \check{\mathbf{x}}(\tau_0) \\ \vdots \\ \check{\mathbf{x}}(\tau_J) \end{bmatrix}, \\ \mathbf{y} &= \begin{bmatrix} \mathbf{y}_0 \\ \vdots \\ \mathbf{y}_K \end{bmatrix}, \quad \mathbf{C} = \text{diag}(\mathbf{C}_0, \dots, \mathbf{C}_K), \quad \mathbf{R} = \text{diag}(\mathbf{R}_0, \dots, \mathbf{R}_K), \\ \check{\mathbf{P}} &= [\check{\mathbf{P}}(t_i, t_j)]_{ij}, \quad \check{\mathbf{P}}_\tau = [\check{\mathbf{P}}(\tau_i, t_j)]_{ij}, \quad \check{\mathbf{P}}_{\tau\tau} = [\check{\mathbf{P}}(\tau_i, \tau_j)]_{ij}. \end{aligned}$$

In GP regression, the matrix, $\check{\mathbf{P}}$, is known as the *kernel matrix*. Based on the factoring discussed in Section 2.2.3, we then have that

$$\begin{aligned} p(\mathbf{x}_\tau | \mathbf{y}) &= \mathcal{N}\left(\underbrace{\check{\mathbf{x}}_\tau + \check{\mathbf{P}}_\tau \mathbf{C}^T (\mathbf{C}\check{\mathbf{P}}\mathbf{C}^T + \mathbf{R})^{-1}(\mathbf{y} - \mathbf{C}\check{\mathbf{x}})}_{\hat{\mathbf{x}}_\tau, \text{ mean}}, \right. \\ &\quad \left. \underbrace{\check{\mathbf{P}}_{\tau\tau} - \check{\mathbf{P}}_\tau \mathbf{C}^T (\mathbf{C}\check{\mathbf{P}}\mathbf{C}^T + \mathbf{R})^{-1} \mathbf{C}\check{\mathbf{P}}_\tau^T}_{\hat{\mathbf{P}}_{\tau\tau}, \text{ covariance}}\right), \quad (3.149) \end{aligned}$$

for the density of the predicted state at the query times, given the measurements.

The expression simplifies further if we take the query times to be exactly the same as the measurement times (i.e., $\tau_k = t_k, K = J$). This implies that

$$\check{\mathbf{P}} = \check{\mathbf{P}}_\tau = \check{\mathbf{P}}_{\tau\tau}, \quad (3.150)$$

and then we can write

$$\begin{aligned} p(\mathbf{x} | \mathbf{y}) &= \mathcal{N}\left(\underbrace{\check{\mathbf{x}} + \check{\mathbf{P}} \mathbf{C}^T (\mathbf{C}\check{\mathbf{P}}\mathbf{C}^T + \mathbf{R})^{-1}(\mathbf{y} - \mathbf{C}\check{\mathbf{x}})}_{\hat{\mathbf{x}}, \text{ mean}}, \right. \\ &\quad \left. \underbrace{\check{\mathbf{P}} - \check{\mathbf{P}} \mathbf{C}^T (\mathbf{C}\check{\mathbf{P}}\mathbf{C}^T + \mathbf{R})^{-1} \mathbf{C}\check{\mathbf{P}}^T}_{\hat{\mathbf{P}}, \text{ covariance}}\right). \quad (3.151) \end{aligned}$$

Or, after an application of the SMW identity in (2.75), we can write

this as

$$p(\mathbf{x}|\mathbf{y}) = \mathcal{N}\left(\underbrace{(\check{\mathbf{P}}^{-1} + \mathbf{C}^T \mathbf{R}^{-1} \mathbf{C})^{-1} (\check{\mathbf{P}}^{-1} \check{\mathbf{x}} + \mathbf{C}^T \mathbf{R}^{-1} \mathbf{y})}_{\hat{\mathbf{x}}, \text{ mean}}, \underbrace{(\check{\mathbf{P}}^{-1} + \mathbf{C}^T \mathbf{R}^{-1} \mathbf{C})^{-1}}_{\hat{\mathbf{P}}, \text{ covariance}}\right). \quad (3.152)$$

Rearranging the mean expression, we have a linear system for $\hat{\mathbf{x}}$:

$$(\check{\mathbf{P}}^{-1} + \mathbf{C}^T \mathbf{R}^{-1} \mathbf{C}) \hat{\mathbf{x}} = \check{\mathbf{P}}^{-1} \check{\mathbf{x}} + \mathbf{C}^T \mathbf{R}^{-1} \mathbf{y}. \quad (3.153)$$

This can be viewed as the solution to the following optimization problem:

$$\hat{\mathbf{x}} = \arg \min_{\mathbf{x}} \frac{1}{2} (\check{\mathbf{x}} - \mathbf{x})^T \check{\mathbf{P}}^{-1} (\check{\mathbf{x}} - \mathbf{x}) + \frac{1}{2} (\mathbf{y} - \mathbf{C} \mathbf{x})^T \mathbf{R}^{-1} (\mathbf{y} - \mathbf{C} \mathbf{x}). \quad (3.154)$$

Note that at implementation we must be careful to use \mathbf{y}_{meas} , the actual measurements received from our sensors.

If, after solving for the estimate at the measurement times, we later want to query the state at some other times of interest ($\tau_0 < \tau_1 < \dots < \tau_J$), we can use the GP interpolation equations to do so:

$$\hat{\mathbf{x}}_\tau = \check{\mathbf{x}}_\tau + (\check{\mathbf{P}}_\tau \check{\mathbf{P}}^{-1}) (\hat{\mathbf{x}} - \check{\mathbf{x}}), \quad (3.155a)$$

$$\hat{\mathbf{P}}_{\tau\tau} = \check{\mathbf{P}}_{\tau\tau} + (\check{\mathbf{P}}_\tau \check{\mathbf{P}}^{-1}) (\hat{\mathbf{P}} - \check{\mathbf{P}}) (\check{\mathbf{P}}_\tau \check{\mathbf{P}}^{-1})^T. \quad (3.155b)$$

This is linear interpolation in the state variable (but not necessarily in time). To arrive at these interpolation equations, we return to (3.151) and rearrange both the mean and covariance expressions:

$$\check{\mathbf{P}}^{-1} (\hat{\mathbf{x}} - \check{\mathbf{x}}) = \mathbf{C}^T (\check{\mathbf{P}} \mathbf{C}^T + \mathbf{R})^{-1} (\mathbf{y} - \mathbf{C} \check{\mathbf{x}}), \quad (3.156a)$$

$$\check{\mathbf{P}}^{-1} (\hat{\mathbf{P}} - \check{\mathbf{P}}) \check{\mathbf{P}}^{-T} = -\mathbf{C}^T (\check{\mathbf{P}} \mathbf{C}^T + \mathbf{R})^{-1} \mathbf{C}. \quad (3.156b)$$

These can then be substituted back into (3.149),

$$\hat{\mathbf{x}}_\tau = \check{\mathbf{x}}_\tau + \check{\mathbf{P}}_\tau \underbrace{\mathbf{C}^T (\check{\mathbf{P}} \mathbf{C}^T + \mathbf{R})^{-1} (\mathbf{y} - \mathbf{C} \check{\mathbf{x}})}_{\check{\mathbf{P}}^{-1} (\hat{\mathbf{x}} - \check{\mathbf{x}})}, \quad (3.157a)$$

$$\hat{\mathbf{P}}_{\tau\tau} = \check{\mathbf{P}}_{\tau\tau} - \check{\mathbf{P}}_\tau \underbrace{\mathbf{C}^T (\check{\mathbf{P}} \mathbf{C}^T + \mathbf{R})^{-1} \mathbf{C}}_{-\check{\mathbf{P}}^{-1} (\hat{\mathbf{P}} - \check{\mathbf{P}}) \check{\mathbf{P}}^{-T}} \check{\mathbf{P}}_\tau^T, \quad (3.157b)$$

to produce (3.155).

In general, this GP approach has complexity $O(K^3 + K^2 J)$ since the initial solve is $O(K^3)$ and the query is $O(K^2 J)$; this is quite expensive, and next we will seek to improve the cost by exploiting the structure of the matrices involved.

3.4.2 A Class of Exactly Sparse Gaussian Process Priors

Next, we will develop a special class of GP priors that lead to very efficient implementation. These priors are based on *linear time-varying (LTV) stochastic differential equations (SDEs)*:

$$\dot{\mathbf{x}}(t) = \mathbf{A}(t)\mathbf{x}(t) + \mathbf{v}(t) + \mathbf{L}(t)\mathbf{w}(t), \quad (3.158)$$

with

$$\mathbf{w}(t) \sim \mathcal{GP}(\mathbf{0}, \mathbf{Q} \delta(t - t')), \quad (3.159)$$

a (stationary) zero-mean GP with (symmetric, positive-definite) *power spectral density matrix*, \mathbf{Q} . In what follows, we will use an engineering approach that avoids introducing *Itō calculus*; we hope that what our treatment lacks in formality it makes up for in accessibility. For a more formal treatment of stochastic differential equations in estimation, see Särkkä (2006).

The general solution to this LTV ordinary differential equation is

$$\mathbf{x}(t) = \Phi(t, t_0)\mathbf{x}(t_0) + \int_{t_0}^t \Phi(t, s)(\mathbf{v}(s) + \mathbf{L}(s)\mathbf{w}(s)) ds, \quad (3.160)$$

where $\Phi(t, s)$ is known as the *transition function* and has the following properties:

$$\Phi(t, t) = \mathbf{1}, \quad (3.161)$$

$$\dot{\Phi}(t, s) = \mathbf{A}(t)\Phi(t, s), \quad (3.162)$$

$$\Phi(t, s) = \Phi(t, r)\Phi(r, s). \quad (3.163)$$

Kiyoshi Itō (1915-2008) was a Japanese mathematician who pioneered the theory of stochastic integration and stochastic differential equations, now known as the *Itō calculus*.

It is usually straightforward to work out the transition function for systems in practice, but there is no general formula.

Mean Function

For the mean function we have

$$\underbrace{E[\mathbf{x}(t)]}_{\check{\mathbf{x}}(t)} = \Phi(t, t_0) \underbrace{E[\mathbf{x}(t_0)]}_{\check{\mathbf{x}}_0} + \int_{t_0}^t \Phi(t, s) \left(\mathbf{v}(s) + \mathbf{L}(s) \underbrace{E[\mathbf{w}(s)]}_{\mathbf{0}} \right) ds, \quad (3.164)$$

where $\check{\mathbf{x}}_0$ is the initial value of the mean at t_0 . Thus, the mean function is

$$\check{\mathbf{x}}(t) = \Phi(t, t_0)\check{\mathbf{x}}_0 + \int_{t_0}^t \Phi(t, s)\mathbf{v}(s) ds. \quad (3.165)$$

If we now have a sequence of times, $t_0 < t_1 < t_2 < \dots < t_K$, then we can write the mean at these times as

$$\check{\mathbf{x}}(t_k) = \Phi(t_k, t_0)\check{\mathbf{x}}_0 + \sum_{n=1}^k \Phi(t_k, t_n)\mathbf{v}_n, \quad (3.166)$$

where

$$\mathbf{v}_k = \int_{t_{k-1}}^{t_k} \Phi(t_k, s) \mathbf{v}(s) ds, \quad k = 1 \dots K. \quad (3.167)$$

Or we can write our system in *lifted form*,

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{v}, \quad (3.168)$$

where

$$\begin{aligned} \dot{\mathbf{x}} &= \begin{bmatrix} \dot{\mathbf{x}}(t_0) \\ \dot{\mathbf{x}}(t_1) \\ \vdots \\ \dot{\mathbf{x}}(t_K) \end{bmatrix}, \quad \mathbf{v} = \begin{bmatrix} \dot{\mathbf{x}}_0 \\ \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_K \end{bmatrix}, \\ \mathbf{A} &= \begin{bmatrix} \mathbf{1} & & & & & \\ \Phi(t_1, t_0) & \mathbf{1} & & & & \\ \Phi(t_2, t_0) & \Phi(t_2, t_1) & \mathbf{1} & & & \\ \vdots & \vdots & \vdots & \ddots & & \\ \Phi(t_{K-1}, t_0) & \Phi(t_{K-1}, t_1) & \Phi(t_{K-1}, t_2) & \cdots & \mathbf{1} & \\ \Phi(t_K, t_0) & \Phi(t_K, t_1) & \Phi(t_K, t_2) & \cdots & \Phi(t_K, t_{K-1}) & \mathbf{1} \end{bmatrix}. \end{aligned} \quad (3.169)$$

Notably, \mathbf{A} , the *lifted transition matrix*, is lower-triangular.

If we assume that $\mathbf{v}(t) = \mathbf{B}(t)\mathbf{u}(t)$ with $\mathbf{u}(t)$ constant between measurement times, we can further simplify the expression. Let \mathbf{u}_k be the constant input when $t \in (t_{k-1}, t_k]$. Then we can define

$$\mathbf{B} = \text{diag}(\mathbf{1}, \mathbf{B}_1, \dots, \mathbf{B}_K), \quad \mathbf{u} = \begin{bmatrix} \dot{\mathbf{x}}_0 \\ \mathbf{u}_1 \\ \vdots \\ \mathbf{u}_K \end{bmatrix}, \quad (3.170)$$

and

$$\mathbf{B}_k = \int_{t_{k-1}}^{t_k} \Phi(t_k, s) \mathbf{B}(s) ds, \quad k = 1 \dots K. \quad (3.171)$$

This allows us to write

$$\dot{\mathbf{x}}(t_k) = \Phi(t_k, t_{k-1}) \dot{\mathbf{x}}(t_{k-1}) + \mathbf{B}_k \mathbf{u}_k, \quad (3.172)$$

and

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{B}\mathbf{u}, \quad (3.173)$$

for the vector of means.

Covariance Function

For the covariance function we have

$$\begin{aligned} & \underbrace{E[(\mathbf{x}(t) - E[\mathbf{x}(t)])(\mathbf{x}(t') - E[\mathbf{x}(t')])^T]}_{\check{\mathbf{P}}(t,t')} \\ &= \Phi(t, t_0) \underbrace{E[(\mathbf{x}(t_0) - E[\mathbf{x}(t_0)])(\mathbf{x}(t_0) - E[\mathbf{x}(t_0)])^T]}_{\check{\mathbf{P}}_0} \Phi(t', t_0)^T \\ &+ \int_{t_0}^t \int_{t_0}^{t'} \Phi(t, s) \mathbf{L}(s) \underbrace{E[\mathbf{w}(s)\mathbf{w}(s')^T]}_{\mathbf{Q}\delta(s-s')} \mathbf{L}(s')^T \Phi(t', s')^T ds' ds, \quad (3.174) \end{aligned}$$

where $\check{\mathbf{P}}_0$ is the initial covariance at t_0 and we have made the assumption that $E[\mathbf{x}(t_0)\mathbf{w}(t)^T] = \mathbf{0}$. Putting this together, we have the following expression for the covariance:

$$\begin{aligned} \check{\mathbf{P}}(t, t') &= \Phi(t, t_0) \check{\mathbf{P}}_0 \Phi(t', t_0)^T \\ &+ \int_{t_0}^t \int_{t_0}^{t'} \Phi(t, s) \mathbf{L}(s) \mathbf{Q} \mathbf{L}(s)^T \Phi(t', s')^T \delta(s - s') ds' ds. \quad (3.175) \end{aligned}$$

Focusing on the second term, we integrate once to see that it is

$$\int_{t_0}^t \Phi(t, s) \mathbf{L}(s) \mathbf{Q} \mathbf{L}(s)^T \Phi(t', s)^T H(t' - s) ds, \quad (3.176)$$

where $H(\cdot)$ is the *Heaviside step function*. There are now three cases to worry about: $t < t'$, $t = t'$, and $t > t'$. In the first case, the upper integration limit terminates the integration, while in the last, the Heaviside step function does the same job. The result is that the second term in the covariance function can be written as

$$\begin{aligned} & \int_{t_0}^{\min(t,t')} \Phi(t, s) \mathbf{L}(s) \mathbf{Q} \mathbf{L}(s)^T \Phi(t', s)^T ds \\ &= \begin{cases} \Phi(t, t') \left(\int_{t_0}^{t'} \Phi(t', s) \mathbf{L}(s) \mathbf{Q} \mathbf{L}(s)^T \Phi(t', s)^T ds \right) & t' < t \\ \int_{t_0}^t \Phi(t, s) \mathbf{L}(s) \mathbf{Q} \mathbf{L}(s)^T \Phi(t, s)^T ds & t = t' \\ \left(\int_{t_0}^t \Phi(t, s) \mathbf{L}(s) \mathbf{Q} \mathbf{L}(s)^T \Phi(t, s)^T ds \right) \Phi(t', t)^T & t < t' \end{cases}. \quad (3.177) \end{aligned}$$

If we now have a sequence of times, $t_0 < t_1 < t_2 < \dots < t_K$, then we can write the covariance between two of these times as

$$\check{\mathbf{P}}(t_i, t_j) = \begin{cases} \Phi(t_i, t_j) \left(\sum_{n=0}^j \Phi(t_j, t_n) \mathbf{Q}_n \Phi(t_j, t_n)^T \right) & t_j < t_i \\ \sum_{n=0}^i \Phi(t_i, t_n) \mathbf{Q}_n \Phi(t_i, t_n)^T & t_i = t_j \\ \left(\sum_{n=0}^i \Phi(t_i, t_n) \mathbf{Q}_n \Phi(t_i, t_n)^T \right) \Phi(t_j, t_i)^T & t_i < t_j \end{cases}, \quad (3.178)$$

where

$$\mathbf{Q}_k = \int_{t_{k-1}}^{t_k} \Phi(t_k, s) \mathbf{L}(s) \mathbf{Q} \mathbf{L}(s)^T \Phi(t_k, s)^T ds, \quad k = 1 \dots K, \quad (3.179)$$

and we let $\mathbf{Q}_0 = \check{\mathbf{P}}_0$ to keep the notation in (3.177) clean.

Given this preparation, we are now ready to state the main result of this section. Let $t_0 < t_1 < t_2 < \dots < t_K$ be a monotonically increasing sequence of time values. Define the *kernel matrix* to be

$$\check{\mathbf{P}} = \left[\begin{array}{c} \Phi(t_i, t_0) \check{\mathbf{P}}_0 \Phi(t_j, t_0)^T \\ + \int_{t_0}^{\min(t_i, t_j)} \Phi(t_i, s) \mathbf{L}(s) \mathbf{Q} \mathbf{L}(s)^T \Phi(t_j, s)^T ds \end{array} \right]_{ij}, \quad (3.180)$$

where $\mathbf{Q} > 0$ is symmetric. Note that $\check{\mathbf{P}}$ has $(K+1) \times (K+1)$ blocks. Then, we can factor $\check{\mathbf{P}}$ according to a block-lower-diagonal-upper decomposition:

$$\check{\mathbf{P}} = \mathbf{A} \mathbf{Q} \mathbf{A}^T, \quad (3.181)$$

where \mathbf{A} is the lower-triangular matrix given in (3.169) and

$$\mathbf{Q}_k = \int_{t_{k-1}}^{t_k} \Phi(t_k, s) \mathbf{L}(s) \mathbf{Q} \mathbf{L}(s)^T \Phi(t_k, s)^T ds, \quad k = 1 \dots K, \quad (3.182)$$

$$\mathbf{Q} = \text{diag}(\check{\mathbf{P}}_0, \mathbf{Q}_1, \mathbf{Q}_2, \dots, \mathbf{Q}_K). \quad (3.183)$$

It follows that $\check{\mathbf{P}}^{-1}$ is block-tridiagonal and is given by

$$\check{\mathbf{P}}^{-1} = (\mathbf{A} \mathbf{Q} \mathbf{A}^T)^{-1} = \mathbf{A}^{-T} \mathbf{Q}^{-1} \mathbf{A}^{-1}, \quad (3.184)$$

where

$$\mathbf{A}^{-1} = \begin{bmatrix} \mathbf{1} & & & & & \\ -\Phi(t_1, t_0) & \mathbf{1} & & & & \\ & -\Phi(t_2, t_1) & \mathbf{1} & & & \\ & & -\Phi(t_3, t_2) & \ddots & & \\ & & & \ddots & \mathbf{1} & \\ & & & & -\Phi(t_K, t_{K-1}) & \mathbf{1} \end{bmatrix}. \quad (3.185)$$

Since \mathbf{A}^{-1} has only the main diagonal and the one below non-zero, and \mathbf{Q}^{-1} is block-diagonal, the block-tridiagonal structure of $\check{\mathbf{P}}^{-1}$ can be verified by carrying out the multiplication. This is precisely the structure we had at the start of the chapter for the batch discrete-time case.

Summary of Prior

We can write our final GP for $\mathbf{x}(t)$ as

$$\begin{aligned} \mathbf{x}(t) \sim \mathcal{GP} & \left(\underbrace{\Phi(t, t_0) \check{\mathbf{x}}_0 + \int_{t_0}^t \Phi(t, s) \mathbf{v}(s) ds}_{\check{\mathbf{x}}(t)}, \right. \\ & \left. \underbrace{\Phi(t, t_0) \check{\mathbf{P}}_0 \Phi(t', t_0)^T + \int_{t_0}^{\min(t, t')} \Phi(t, s) \mathbf{L}(s) \mathbf{Q} \mathbf{L}(s)^T \Phi(t', s)^T ds}_{\check{\mathbf{P}}(t, t')} \right). \end{aligned} \quad (3.186)$$

At the measurement times, $t_0 < t_1 < \dots < t_K$, we can also then write

$$\mathbf{x} \sim \mathcal{N}(\check{\mathbf{x}}, \check{\mathbf{P}}) = \mathcal{N}(\mathbf{A}\mathbf{v}, \mathbf{A}\mathbf{Q}\mathbf{A}^T), \quad (3.187)$$

and we can further substitute $\mathbf{v} = \mathbf{B}\mathbf{u}$ in the case that the inputs are constant between measurement times.

Querying the GP

As discussed above, if we solve for the trajectory at the measurement times, we may want to query it at other times of interest as well. This can be done through the GP linear interpolation equations in (3.155). Without loss of generality, we consider a single query time, $t_k \leq \tau < t_{k+1}$, and so in this case we write

$$\hat{\mathbf{x}}(\tau) = \check{\mathbf{x}}(\tau) + \check{\mathbf{P}}(\tau) \check{\mathbf{P}}^{-1}(\hat{\mathbf{x}} - \check{\mathbf{x}}), \quad (3.188a)$$

$$\hat{\mathbf{P}}(\tau, \tau) = \check{\mathbf{P}}(\tau, \tau) + \check{\mathbf{P}}(\tau) \check{\mathbf{P}}^{-1} (\hat{\mathbf{P}} - \check{\mathbf{P}}) \check{\mathbf{P}}^{-T} \check{\mathbf{P}}(\tau)^T. \quad (3.188b)$$

For the mean function at the query time, we simply have

$$\check{\mathbf{x}}(\tau) = \Phi(\tau, t_k) \check{\mathbf{x}}(t_k) + \int_{t_k}^{\tau} \Phi(\tau, s) \mathbf{v}(s) ds, \quad (3.189)$$

which has complexity $O(1)$ to evaluate. For the covariance function at the query time we have

$$\check{\mathbf{P}}(\tau, \tau) = \Phi(\tau, t_k) \check{\mathbf{P}}(t_k, t_k) \Phi(\tau, t_k)^T + \int_{t_k}^{\tau} \Phi(\tau, s) \mathbf{L}(s) \mathbf{Q} \mathbf{L}(s)^T \Phi(\tau, s)^T ds, \quad (3.190)$$

which is also $O(1)$ to evaluate.

We now examine the sparsity of the product $\check{\mathbf{P}}(\tau) \check{\mathbf{P}}^{-1}$ in the case of a general LTV process model. The matrix, $\check{\mathbf{P}}(\tau)$, can be written as

$$\check{\mathbf{P}}(\tau) = [\check{\mathbf{P}}(\tau, t_0) \quad \check{\mathbf{P}}(\tau, t_1) \quad \dots \quad \check{\mathbf{P}}(\tau, t_K)]. \quad (3.191)$$

The individual blocks are given by

$$\check{\mathbf{P}}(\tau, t_j) = \begin{cases} \Phi(\tau, t_k) \Phi(t_k, t_j) \left(\sum_{n=0}^j \Phi(t_j, t_n) \mathbf{Q}_n \Phi(t_j, t_n)^T \right) & t_j < t_k \\ \Phi(\tau, t_k) \left(\sum_{n=0}^k \Phi(t_k, t_n) \mathbf{Q}_n \Phi(t_k, t_n)^T \right) & t_k = t_j \\ \Phi(\tau, t_k) \left(\sum_{n=0}^k \Phi(t_k, t_n) \mathbf{Q}_n \Phi(t_k, t_n)^T \right) \Phi(t_{k+1}, t_k)^T \\ + \mathbf{Q}_\tau \Phi(t_{k+1}, \tau)^T & t_{k+1} = t_j \\ \Phi(\tau, t_k) \left(\sum_{n=0}^k \Phi(t_k, t_n) \mathbf{Q}_n \Phi(t_k, t_n)^T \right) \Phi(t_j, t_k)^T \\ + \mathbf{Q}_\tau \Phi(t_{k+1}, \tau)^T \Phi(t_j, t_{k+1})^T & t_{k+1} < t_j \end{cases}, \quad (3.192)$$

where

$$\mathbf{Q}_\tau = \int_{t_k}^\tau \Phi(\tau, s) \mathbf{L}(s) \mathbf{Q} \mathbf{L}(s)^T \Phi(\tau, s)^T ds. \quad (3.193)$$

Although this looks difficult to work with, we may write

$$\check{\mathbf{P}}(\tau) = \mathbf{V}(\tau) \mathbf{A}^T, \quad (3.194)$$

where \mathbf{A} was defined before and

$$\mathbf{V}(\tau) = \begin{bmatrix} \Phi(\tau, t_k) \Phi(t_k, t_0) \check{\mathbf{P}}_0 & \Phi(\tau, t_k) \Phi(t_k, t_1) \mathbf{Q}_1 & \cdots \\ \cdots & \Phi(\tau, t_k) \Phi(t_k, t_{k-1}) \mathbf{Q}_{k-1} & \Phi(\tau, t_k) \mathbf{Q}_k & \mathbf{Q}_\tau \Phi(t_{k+1}, \tau)^T & \cdots \\ & & & \cdots & \mathbf{0} & \cdots & \mathbf{0} \end{bmatrix}. \quad (3.195)$$

Returning to the desired product, we have

$$\check{\mathbf{P}}(\tau) \check{\mathbf{P}}^{-1} = \mathbf{V}(\tau) \underbrace{\mathbf{A}^T \mathbf{A}^{-T}}_1 \mathbf{Q}^{-1} \mathbf{A}^{-1} = \mathbf{V}(\tau) \mathbf{Q}^{-1} \mathbf{A}^{-1}. \quad (3.196)$$

Since \mathbf{Q}^{-1} is block-diagonal and \mathbf{A}^{-1} has only the main diagonal and the one below it non-zero, we can evaluate the product very efficiently. Working it out, we have

$$\check{\mathbf{P}}(\tau) \check{\mathbf{P}}^{-1} = \begin{bmatrix} \mathbf{0} & \cdots & \mathbf{0} & \underbrace{\Phi(\tau, t_k) - \mathbf{Q}_\tau \Phi(t_{k+1}, \tau)^T \mathbf{Q}_{k+1}^{-1} \Phi(t_{k+1}, t_k)}_{\Lambda(\tau), \text{ block column } k} \\ \cdots & & & \underbrace{\mathbf{Q}_\tau \Phi(t_{k+1}, \tau)^T \mathbf{Q}_{k+1}^{-1}}_{\Psi(\tau), \text{ block column } k+1} & \mathbf{0} & \cdots & \mathbf{0} \end{bmatrix}, \quad (3.197)$$

which has exactly two non-zero block columns. Inserting this into (3.188), we have

$$\hat{\mathbf{x}}(\tau) = \check{\mathbf{x}}(\tau) + [\Lambda(\tau) \quad \Psi(\tau)] \left(\begin{bmatrix} \hat{\mathbf{x}}_k \\ \hat{\mathbf{x}}_{k+1} \end{bmatrix} - \begin{bmatrix} \check{\mathbf{x}}(t_k) \\ \check{\mathbf{x}}(t_{k+1}) \end{bmatrix} \right), \quad (3.198a)$$

$$\begin{aligned} \hat{\mathbf{P}}(\tau, \tau) &= \check{\mathbf{P}}(\tau, \tau) + [\Lambda(\tau) \quad \Psi(\tau)] \left(\begin{bmatrix} \hat{\mathbf{P}}_{k,k} & \hat{\mathbf{P}}_{k,k+1} \\ \hat{\mathbf{P}}_{k+1,k} & \hat{\mathbf{P}}_{k+1,k+1} \end{bmatrix} \right. \\ &\quad \left. - \begin{bmatrix} \check{\mathbf{P}}(t_k, t_k) & \check{\mathbf{P}}(t_k, t_{k+1}) \\ \check{\mathbf{P}}(t_{k+1}, t_k) & \check{\mathbf{P}}(t_{k+1}, t_{k+1}) \end{bmatrix} \right) \begin{bmatrix} \Lambda(\tau)^T \\ \Psi(\tau)^T \end{bmatrix}, \end{aligned} \quad (3.198b)$$

which is a simple combination of just the two terms from t_k and t_{k+1} . Thus, to query the trajectory at a single time of interest is $O(1)$ complexity.

Example 3.1 As a simple example, consider the system

$$\dot{\mathbf{x}}(t) = \mathbf{w}(t), \quad (3.199)$$

which can be written as

$$\dot{\mathbf{x}}(t) = \mathbf{A}(t)\mathbf{x}(t) + \mathbf{v}(t) + \mathbf{L}(t)\mathbf{w}(t), \quad (3.200)$$

with $\mathbf{A}(t) = \mathbf{0}$, $\mathbf{v}(t) = \mathbf{0}$, $\mathbf{L}(t) = \mathbf{1}$. In this case, the query equation becomes

$$\hat{\mathbf{x}}_\tau = (1 - \alpha)\hat{\mathbf{x}}_k + \alpha\hat{\mathbf{x}}_{k+1}, \quad (3.201)$$

assuming the mean function is zero everywhere and where

$$\alpha = \frac{\tau - t_k}{t_{k+1} - t_k} \in [0, 1], \quad (3.202)$$

which is a familiar interpolation scheme that is linear in τ . More complicated process models lead to more complicated interpolation equations.

3.4.3 Linear Time-Invariant Case

Naturally, the equations simplify considerably in the *linear time-invariant (LTI)* case:

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) + \mathbf{L}\mathbf{w}(t), \quad (3.203)$$

with \mathbf{A} , \mathbf{B} , and \mathbf{L} constant²². The transition function is simply

$$\Phi(t, s) = \exp(\mathbf{A}(t - s)), \quad (3.204)$$

which we note depends only on the difference of the two times (i.e., it is stationary). We can therefore write,

$$\Delta t_{k:k-1} = t_k - t_{k-1}, \quad k = 1 \dots K, \quad (3.205)$$

$$\Phi(t_k, t_{k-1}) = \exp(\mathbf{A} \Delta t_{k:k-1}), \quad k = 1 \dots K, \quad (3.206)$$

$$\Phi(t_k, t_j) = \Phi(t_k, t_{k-1})\Phi(t_{k-1}, t_{k-2}) \cdots \Phi(t_{j+1}, t_j), \quad (3.207)$$

to simplify matters.

²² We use italicized symbols for the time-invariant system matrices to avoid confusion with the lifted-form quantities.

Mean Function

For the mean function we have the following simplification:

$$\mathbf{v}_k = \int_0^{\Delta t_{k:k-1}} \exp(\mathbf{A}(\Delta t_{k:k-1} - s)) \mathbf{B}\mathbf{u}(s) ds, \quad k = 1 \dots K. \quad (3.208)$$

If we assume that $\mathbf{u}(t)$ is constant between each pair of consecutive measurement times, we can further simplify the expression. Let \mathbf{u}_k be the constant input when $t \in (t_{k-1}, t_k]$. Then we can define

$$\mathbf{B} = \text{diag}(\mathbf{1}, \mathbf{B}_1, \dots, \mathbf{B}_M), \quad \mathbf{u} = \begin{bmatrix} \check{\mathbf{x}}_0 \\ \mathbf{u}_1 \\ \vdots \\ \mathbf{u}_M \end{bmatrix}, \quad (3.209)$$

and

$$\begin{aligned} \mathbf{B}_k &= \int_0^{\Delta t_{k:k-1}} \exp(\mathbf{A}(\Delta t_{k:k-1} - s)) ds \mathbf{B} \\ &= \Phi(t_k, t_{k-1}) (\mathbf{1} - \Phi(t_k, t_{k-1})^{-1}) \mathbf{A}^{-1} \mathbf{B}, \quad k = 1 \dots K. \end{aligned} \quad (3.210)$$

This allows us to write

$$\check{\mathbf{x}} = \mathbf{ABu} \quad (3.211)$$

for the vector of means.

Covariance Function

For the covariance function, we have the simplification

$$\mathbf{Q}_k = \int_0^{\Delta t_{k:k-1}} \exp(\mathbf{A}(\Delta t_{k:k-1} - s)) \mathbf{L} \mathbf{Q} \mathbf{L}^T \exp(\mathbf{A}(\Delta t_{k:k-1} - s))^T ds \quad (3.212)$$

for $k = 1 \dots K$. This is relatively straightforward to evaluate, particularly if \mathbf{A} is nilpotent. Letting

$$\mathbf{Q} = \text{diag}(\check{\mathbf{P}}_0, \mathbf{Q}_1, \mathbf{Q}_2, \dots, \mathbf{Q}_K), \quad (3.213)$$

we then have

$$\check{\mathbf{P}} = \mathbf{AQ} \mathbf{A}^T, \quad (3.214)$$

for the covariance matrix.

Querying the GP

To query the GP, we need the following quantities:

$$\Phi(t_{k+1}, \tau) = \exp(\mathbf{A} \Delta t_{k+1:\tau}), \quad \Delta t_{k+1:\tau} = t_{k+1} - \tau, \quad (3.215)$$

$$\Phi(\tau, t_k) = \exp(\mathbf{A} \Delta t_{\tau:k}), \quad \Delta t_{\tau:k} = \tau - t_k, \quad (3.216)$$

$$\mathbf{Q}_\tau = \int_0^{\Delta t_{\tau:k}} \exp(\mathbf{A}(\Delta t_{\tau:k} - s)) \mathbf{L} \mathbf{Q} \mathbf{L}^T \exp(\mathbf{A}(\Delta t_{\tau:k} - s)^T) ds. \quad (3.217)$$

Our interpolation equation is still

$$\hat{\mathbf{x}}(\tau) = \check{\mathbf{x}}(\tau) + (\Phi(\tau, t_k) - \mathbf{Q}_\tau \Phi(t_{k+1}, \tau)^T \mathbf{Q}_{k+1}^{-1} \Phi(t_{k+1}, t_k)) (\hat{\mathbf{x}}_k - \check{\mathbf{x}}_k) \\ + \mathbf{Q}_\tau \Phi(t_{k+1}, \tau)^T \mathbf{Q}_{k+1}^{-1} (\hat{\mathbf{x}}_{k+1} - \check{\mathbf{x}}_{k+1}), \quad (3.218)$$

which is a linear combination of just the two terms from t_k and t_{k+1} .

Example 3.2 Consider the case

$$\ddot{\mathbf{p}}(t) = \mathbf{w}(t), \quad (3.219)$$

where $\mathbf{p}(t)$ corresponds to position and

$$\mathbf{w}(t) \sim \mathcal{GP}(\mathbf{0}, \mathbf{Q} \delta(t - t')), \quad (3.220)$$

is white noise as before. This corresponds to white noise on acceleration (i.e., the ‘constant velocity’ model). We can cast this in the form

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) + \mathbf{L}\mathbf{w}(t) \quad (3.221)$$

by taking

$$\mathbf{x}(t) = \begin{bmatrix} \mathbf{p}(t) \\ \dot{\mathbf{p}}(t) \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{1} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \quad \mathbf{B} = \mathbf{0}, \quad \mathbf{L} = \begin{bmatrix} \mathbf{0} \\ \mathbf{1} \end{bmatrix}. \quad (3.222)$$

In this case we have

$$\exp(\mathbf{A}\Delta t) = \mathbf{1} + \mathbf{A}\Delta t + \frac{1}{2} \underbrace{\mathbf{A}^2}_{\mathbf{0}} \Delta t^2 + \dots = \mathbf{1} + \begin{bmatrix} \mathbf{0} & \mathbf{1} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \Delta t = \begin{bmatrix} \mathbf{1} & \Delta t \mathbf{1} \\ \mathbf{0} & \mathbf{1} \end{bmatrix}, \quad (3.223)$$

since \mathbf{A} is nilpotent. Therefore, we have

$$\Phi(t_k, t_{k-1}) = \begin{bmatrix} \mathbf{1} & \Delta t_{k:k-1} \mathbf{1} \\ \mathbf{0} & \mathbf{1} \end{bmatrix}. \quad (3.224)$$

For the \mathbf{Q}_k we have

$$\begin{aligned} \mathbf{Q}_k &= \int_0^{\Delta t_{k:k-1}} \begin{bmatrix} \mathbf{1} & (\Delta t_{k:k-1} - s) \mathbf{1} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \begin{bmatrix} \mathbf{0} \\ \mathbf{1} \end{bmatrix} \mathbf{Q} \begin{bmatrix} \mathbf{0} & \mathbf{1} \\ (\Delta t_{k:k-1} - s) \mathbf{1} & \mathbf{1} \end{bmatrix} ds \\ &= \int_0^{\Delta t_{k:k-1}} \begin{bmatrix} (\Delta t_{k:k-1} - s)^2 \mathbf{Q} & (\Delta t_{k:k-1} - s) \mathbf{Q} \\ (\Delta t_{k:k-1} - s) \mathbf{Q} & \mathbf{Q} \end{bmatrix} ds \\ &= \begin{bmatrix} \frac{1}{3} \Delta t_{k:k-1}^3 \mathbf{Q} & \frac{1}{2} \Delta t_{k:k-1}^2 \mathbf{Q} \\ \frac{1}{2} \Delta t_{k:k-1}^2 \mathbf{Q} & \Delta t_{k:k-1} \mathbf{Q} \end{bmatrix}, \end{aligned} \quad (3.225)$$

which we note is positive-definite even though $\mathbf{L}\mathbf{Q}\mathbf{L}^T$ is not. The inverse is

$$\mathbf{Q}_k^{-1} = \begin{bmatrix} 12\Delta t_{k:k-1}^{-3}\mathbf{Q}^{-1} & -6\Delta t_{k:k-1}^{-2}\mathbf{Q}^{-1} \\ -6\Delta t_{k:k-1}^{-2}\mathbf{Q}^{-1} & 4\Delta t_{k:k-1}^{-1}\mathbf{Q}^{-1} \end{bmatrix}, \quad (3.226)$$

which is needed to construct $\check{\mathbf{P}}^{-1}$. For the mean function we have

$$\check{\mathbf{x}}_k = \Phi(t_k, t_0)\check{\mathbf{x}}_0, \quad k = 1 \dots K, \quad (3.227)$$

which can be stacked and written as

$$\check{\mathbf{x}} = \mathbf{A} \begin{bmatrix} \check{\mathbf{x}}_0 \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix} \quad (3.228)$$

for convenience.

For trajectory queries, we also need

$$\begin{aligned} \Phi(\tau, t_k) &= \begin{bmatrix} \mathbf{1} & \Delta t_{\tau:k}\mathbf{1} \\ \mathbf{0} & \mathbf{1} \end{bmatrix}, \quad \Phi(t_{k+1}, \tau) = \begin{bmatrix} \mathbf{1} & \Delta t_{k+1:\tau}\mathbf{1} \\ \mathbf{0} & \mathbf{1} \end{bmatrix}, \\ \check{\mathbf{x}}_\tau &= \Phi(\tau, t_k)\check{\mathbf{x}}_k, \quad \mathbf{Q}_\tau = \begin{bmatrix} \frac{1}{3}\Delta t_{\tau:k}^3\mathbf{Q} & \frac{1}{2}\Delta t_{\tau:k}^2\mathbf{Q} \\ \frac{1}{2}\Delta t_{\tau:k}^2\mathbf{Q} & \Delta t_{\tau:k}\mathbf{Q} \end{bmatrix}, \end{aligned} \quad (3.229)$$

which we see will result in a scheme that is not linear in τ . Substituting these into the interpolation equation, we have

$$\begin{aligned} \hat{\mathbf{x}}_\tau &= \check{\mathbf{x}}_\tau + (\Phi(\tau, t_k) - \mathbf{Q}_\tau \Phi(t_{k+1}, \tau)^T \mathbf{Q}_{k+1}^{-1} \Phi(t_{k+1}, t_k)) (\hat{\mathbf{x}}_k - \check{\mathbf{x}}_k) \\ &\quad + \mathbf{Q}_\tau \Phi(t_{k+1}, \tau)^T \mathbf{Q}_{k+1}^{-1} (\hat{\mathbf{x}}_{k+1} - \check{\mathbf{x}}_{k+1}) \\ &= \check{\mathbf{x}}_\tau + \begin{bmatrix} (1 - 3\alpha^2 + 2\alpha^3)\mathbf{1} & T(\alpha - 2\alpha^2 + \alpha^3)\mathbf{1} \\ \frac{1}{T}6(-\alpha + \alpha^2)\mathbf{1} & (1 - 4\alpha + 3\alpha^2)\mathbf{1} \end{bmatrix} (\hat{\mathbf{x}}_k - \check{\mathbf{x}}_k) \\ &\quad + \begin{bmatrix} (3\alpha^2 - 2\alpha^3)\mathbf{1} & T(-\alpha^2 + \alpha^3)\mathbf{1} \\ \frac{1}{T}6(\alpha - \alpha^2)\mathbf{1} & (-2\alpha + 3\alpha^2)\mathbf{1} \end{bmatrix} (\hat{\mathbf{x}}_{k+1} - \check{\mathbf{x}}_{k+1}), \end{aligned} \quad (3.230)$$

where

$$\alpha = \frac{\tau - t_k}{t_{k+1} - t_k} \in [0, 1], \quad T = \Delta t_{k+1:k} = t_{k+1} - t_k. \quad (3.231)$$

Remarkably, the top row (corresponding to position) is precisely a *cubic Hermite polynomial* interpolation:

$$\begin{aligned} \hat{\mathbf{p}}_\tau - \check{\mathbf{p}}_\tau &= h_{00}(\alpha)(\hat{\mathbf{p}}_k - \check{\mathbf{p}}_k) + h_{10}(\alpha)T(\hat{\mathbf{p}}_k - \check{\mathbf{p}}_k) \\ &\quad + h_{01}(\alpha)(\hat{\mathbf{p}}_{k+1} - \check{\mathbf{p}}_{k+1}) + h_{11}(\alpha)T(\hat{\mathbf{p}}_{k+1} - \check{\mathbf{p}}_{k+1}), \end{aligned} \quad (3.232)$$

where

$$h_{00}(\alpha) = 1 - 3\alpha^2 + 2\alpha^3, \quad h_{10}(\alpha) = \alpha - 2\alpha^2 + \alpha^3, \quad (3.233a)$$

$$h_{01}(\alpha) = 3\alpha^2 - 2\alpha^3, \quad h_{11}(\alpha) = -\alpha^2 + \alpha^3, \quad (3.233b)$$

Charles Hermite (1822-1901) was a French mathematician who did research on a variety of topics, including orthogonal polynomials.

are the *Hermite basis functions*. The bottom row (corresponding to velocity) is only quadratic in α , and the basis functions are the derivatives of the ones used to interpolate position. It is very important to note that this Hermite interpolation scheme arises automatically from using the GP regression approach and our choice of prior motion model. At implementation, we may work directly with the general matrix equations and avoid working out the details of the resulting interpolation scheme.

It is also easy to verify that when $\alpha = 0$, we have

$$\hat{\mathbf{x}}_\tau = \check{\mathbf{x}}_\tau + (\hat{\mathbf{x}}_k - \check{\mathbf{x}}_k), \quad (3.234)$$

and when $\alpha = 1$, we have

$$\hat{\mathbf{x}}_\tau = \check{\mathbf{x}}_\tau + (\hat{\mathbf{x}}_{k+1} - \check{\mathbf{x}}_{k+1}), \quad (3.235)$$

which seem to be sensible boundary conditions.

3.4.4 Relationship to Batch Discrete-Time Estimation

Now that we have seen how to efficiently represent the prior, we can revisit the GP optimization problem described by (3.154). Substituting in our prior terms, the problem becomes

$$\begin{aligned} \hat{\mathbf{x}} = \arg \min_{\mathbf{x}} & \frac{1}{2} \underbrace{(\mathbf{A}\mathbf{v} - \mathbf{x})^T}_{\check{\mathbf{x}}} \underbrace{\mathbf{A}^{-T} \mathbf{Q}^{-1} \mathbf{A}^{-1}}_{\check{\mathbf{P}}^{-1}} \underbrace{(\mathbf{A}\mathbf{v} - \mathbf{x})}_{\check{\mathbf{x}}} \\ & + \frac{1}{2} (\mathbf{y} - \mathbf{Cx})^T \mathbf{R}^{-1} (\mathbf{y} - \mathbf{Cx}). \end{aligned} \quad (3.236)$$

Rearranging, we have

$$\begin{aligned} \hat{\mathbf{x}} = \arg \min_{\mathbf{x}} & \frac{1}{2} (\mathbf{v} - \mathbf{A}^{-1} \mathbf{x})^T \mathbf{Q}^{-1} (\mathbf{v} - \mathbf{A}^{-1} \mathbf{x}) \\ & + \frac{1}{2} (\mathbf{y} - \mathbf{Cx})^T \mathbf{R}^{-1} (\mathbf{y} - \mathbf{Cx}). \end{aligned} \quad (3.237)$$

The solution to this optimization problem is given by

$$\underbrace{(\mathbf{A}^{-T} \mathbf{Q}^{-1} \mathbf{A}^{-1} + \mathbf{C}^T \mathbf{R}^{-1} \mathbf{C})}_{\text{block-tridiagonal}} \hat{\mathbf{x}} = \mathbf{A}^{-T} \mathbf{Q}^{-1} \mathbf{v} + \mathbf{C}^T \mathbf{R}^{-1} \mathbf{y}. \quad (3.238)$$

Because the left-hand side is block-tridiagonal, we can solve this system of equations in $O(K)$ time with a sparse solver (e.g., sparse Cholesky decomposition followed by sparse forward-backward passes). To query the trajectory at J extra times will be $O(J)$ since each query is $O(1)$. This means that we can solve for the state at the measurement and query times in $O(K + J)$ time. This is a big improvement over the $O(K^3 + K^2 J)$ cost when we did not exploit the sparse structure of our particular class of GP priors.

This is identical to the system of equations we had to solve in the discrete-time approach earlier. Thus, the discrete-time approach can *exactly* capture the continuous-time approach (at the measurement times), and both can be viewed as carrying out Gaussian process regression.

3.5 Summary

The main take-away points from this chapter are as follows:

1. When the motion and observation models are linear, and the measurement and process noises are zero-mean Gaussian, the batch and recursive solutions to state estimation are straightforward, requiring no approximation.
2. The Bayesian posterior of a linear-Gaussian estimation problem is *exactly* Gaussian. This implies that the MAP solution is the same as the mean of the full Bayesian solution, since the mode and the mean of a Gaussian are one and the same.
3. The batch, discrete-time, linear-Gaussian solution can exactly implement (at the measurement times) the case where a continuous-time motion model is employed; appropriate prior terms must be used for this to be true.

The next chapter will investigate what happens when the motion and observation models are nonlinear.

3.6 Exercises

- 3.6.1 Consider the discrete-time system,

$$\begin{aligned} x_k &= x_{k-1} + v_k + w_k, & w_k &\sim \mathcal{N}(0, Q), \\ y_k &= x_k + n_k, & n_k &\sim \mathcal{N}(0, R), \end{aligned}$$

which could represent a cart moving back and forth along the x -axis. The initial state, \tilde{x}_0 , is unknown. Set up the system of equations for the batch least-squares estimation approach:

$$(\mathbf{H}^T \mathbf{W}^{-1} \mathbf{H}) \hat{\mathbf{x}} = \mathbf{H}^T \mathbf{W}^{-1} \mathbf{z}.$$

In other words, work out the details of \mathbf{H} , \mathbf{W} , \mathbf{z} , and $\hat{\mathbf{x}}$, for this system. Take the maximum timestep to be $K = 5$. Assume all the noises are uncorrelated with one another. Will a unique solution exist to the problem?

- 3.6.2 Using the same system as the first question, set $Q = R = 1$ and

show that

$$\mathbf{H}^T \mathbf{W}^{-1} \mathbf{H} = \begin{bmatrix} 2 & -1 & & \\ -1 & 3 & -1 & \\ & -1 & 3 & -1 \\ & & -1 & 3 & -1 \\ & & & -1 & 3 & -1 \\ & & & & -1 & 2 \end{bmatrix}.$$

What will be the sparsity pattern of the Cholesky factor, \mathbf{L} , such that $\mathbf{L}\mathbf{L}^T = \mathbf{H}^T \mathbf{W}^{-1} \mathbf{H}$?

- 3.6.3 Using the same system as the first question, modify the least-squares solution for the case in which the measurements noises are correlated with one another in the following way:

$$E[y_k y_\ell] = \begin{cases} R & |k - \ell| = 0 \\ R/2 & |k - \ell| = 1 \\ R/4 & |k - \ell| = 2 \\ 0 & \text{otherwise} \end{cases}.$$

Will a unique least-squares solution still exist?

- 3.6.4 Using the same system as the first question, work out the details of the Kalman filter solution. In this case, assume that the initial conditions for the mean and covariance are \check{x}_0 and \check{P}_0 , respectively. Show that the steady-state values for the prior and posterior covariances, \check{P} and \hat{P} , as $K \rightarrow \infty$ are the solutions to the following quadratics:

$$\begin{aligned} \check{P}^2 - Q\check{P} - QR &= 0, \\ \hat{P}^2 + Q\hat{P} - QR &= 0, \end{aligned}$$

which are two versions of the discrete algebraic Riccati equations. Explain why only one of the two roots to each quadratic is physically possible.

- 3.6.5 Using the MAP approach of Section 3.3.2, derive a version of the Kalman filter that recurses backward in time rather than forward.

3.6.6 Show that

$$\begin{aligned} & \begin{bmatrix} \mathbf{1} \\ \mathbf{A} & \mathbf{1} \\ \mathbf{A}^2 & \mathbf{A} & \mathbf{1} \\ \vdots & \vdots & \vdots & \ddots \\ \mathbf{A}^{K-1} & \mathbf{A}^{K-2} & \mathbf{A}^{K-3} & \cdots & \mathbf{1} \\ \mathbf{A}^K & \mathbf{A}^{K-1} & \mathbf{A}^{K-2} & \cdots & \mathbf{A} & \mathbf{1} \end{bmatrix}^{-1} \\ &= \begin{bmatrix} \mathbf{1} \\ -\mathbf{A} & \mathbf{1} \\ -\mathbf{A} & -\mathbf{A} & \mathbf{1} \\ & -\mathbf{A} & \ddots \\ & \ddots & \mathbf{1} \\ & & -\mathbf{A} & \mathbf{1} \end{bmatrix}. \end{aligned}$$

3.6.7 We have seen that for the batch least-squares solution, the posterior covariance is given by

$$\hat{\mathbf{P}} = (\mathbf{H}^T \mathbf{W}^{-1} \mathbf{H})^{-1}.$$

We have also seen that the computational cost of performing a Cholesky decomposition,

$$\mathbf{L} \mathbf{L}^T = \mathbf{H}^T \mathbf{W}^{-1} \mathbf{H},$$

is $O(N(K + 1))$, owing to the sparsity of the system. Inverting, we have

$$\hat{\mathbf{P}} = \mathbf{L}^{-T} \mathbf{L}^{-1}.$$

Comment on the computational cost of computing $\hat{\mathbf{P}}$ by this approach.

4

Nonlinear Non-Gaussian Estimation

This chapter is one of the most important ones contained in this book. Here we examine how to deal with the fact that in the real world, there are no linear-Gaussian systems. It should be stated up front that *nonlinear, non-Gaussian (NLNG)* estimation is very much still an active research topic. The ideas in this chapter provide only some of the more common approaches to dealing with nonlinear and/or non-Gaussian systems¹. We begin by contrasting full Bayesian to *maximum a posteriori (MAP)* estimation for nonlinear systems. We then introduce a general theoretical framework for recursive filtering problems called the *Bayes filter*. Several of the more common filtering techniques are shown to be approximations of the Bayes filter: extended Kalman filter, sigma-point Kalman filter, particle filter. We then return to batch estimation for nonlinear systems, both in discrete and continuous time. Some books that address nonlinear estimation include Jazwinski (1970), Maybeck (1994), and Simon (2006).

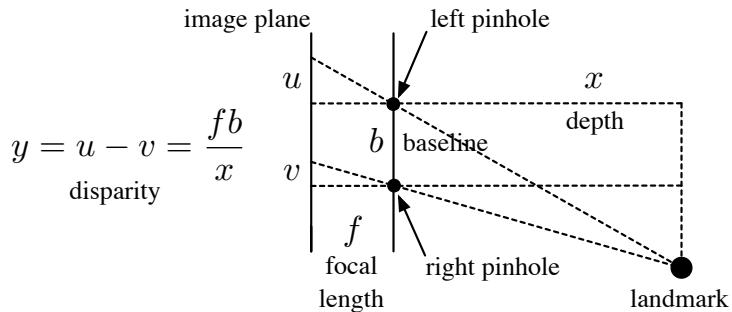
4.1 Introduction

In the linear-Gaussian chapter, we discussed two perspectives to estimation: *full Bayesian* and *maximum a posteriori*. We saw that for linear motion and observation models driven by Gaussian noise, these two paradigms come to the same answer (i.e., the MAP point was the mean of the full Bayesian approach); this is because the full posterior is exactly Gaussian and therefore the mean and mode (i.e., maximum) are the same point.

This is not true once we move to nonlinear models, since the full Bayesian posterior is no longer Gaussian. To provide some intuition on this topic, this section considers a simplified, one-dimensional, nonlinear estimation problem: estimating the position of a landmark from a stereo camera.

¹ Even most of the methods in this chapter actually assume the noise is Gaussian.

Figure 4.1
Idealized stereo camera model relating the landmark depth, x , to the (noise-free) disparity measurement, y .



4.1.1 Full Bayesian Estimation

To gain some intuition, consider a simple estimation problem using a nonlinear, camera model:

$$y = \frac{fb}{x} + n. \quad (4.1)$$

This is the type of nonlinearity present in a *stereo camera* (cf., Figure 4.1), where the state, x , is the position of a landmark (in metres), the measurement, y , is the disparity between the horizontal coordinates of the landmark in the left and right images (in pixels), f is the focal length (in pixels), b is the baseline (horizontal distance between left and right cameras; in metres), and n is the measurement noise (in pixels).

To perform Bayesian inference,

$$p(x|y) = \frac{p(y|x)p(x)}{\int_{-\infty}^{\infty} p(y|x)p(x) dx}, \quad (4.2)$$

we require expressions for $p(y|x)$ and $p(x)$. We meet this requirement by making two assumptions. First, we assume that the measurement noise is zero-mean Gaussian, $n \sim \mathcal{N}(0, R)$, such that

$$p(y|x) = \mathcal{N}\left(\frac{fb}{x}, R\right) = \frac{1}{\sqrt{2\pi R}} \exp\left(-\frac{1}{2R}\left(y - \frac{fb}{x}\right)^2\right), \quad (4.3)$$

and second, we assume that the prior is Gaussian, where

$$p(x) = \mathcal{N}(\check{x}, \check{P}) = \frac{1}{\sqrt{2\pi \check{P}}} \exp\left(-\frac{1}{2\check{P}}(x - \check{x})^2\right). \quad (4.4)$$

Before we continue, we note that the Bayesian framework provides an implied order of operations that we would like to make explicit:

assign prior \rightarrow draw x_{true} \rightarrow draw y_{meas} \rightarrow compute posterior.

In words, we start with a prior. The ‘true’ state is then drawn from the prior, and the measurement is generated by observing the true state through the camera model and adding noise. The estimator then reconstructs the posterior from the measurement and prior, without

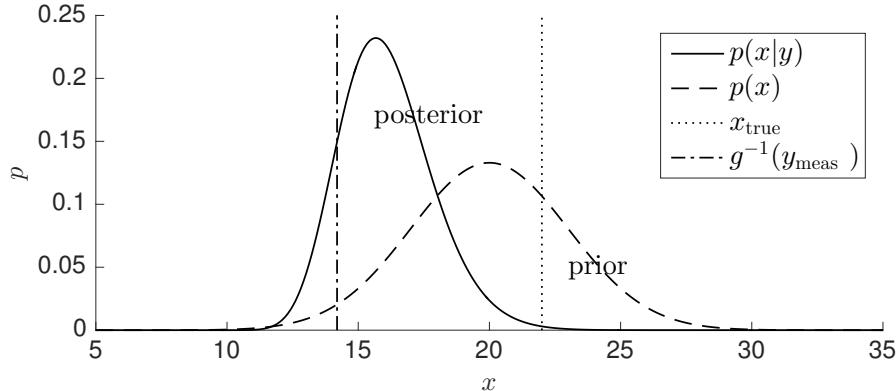


Figure 4.2
 Example of
 Bayesian inference
 on one-dimensional
 stereo camera
 example. We see
 that the full
 posterior is not
 Gaussian, owing to
 the nonlinear
 measurement
 model.

knowing x_{true} . This process is necessary to ensure ‘fair’ comparison between state estimation algorithms.

To put these mathematical models into practical terms, let us assign the following numerical values to the problem:

$$\begin{aligned} \dot{x} &= 20 \text{ [m]}, & \dot{P} &= 9 \text{ [m}^2], \\ f &= 400 \text{ [pixel]}, & b &= 0.1 \text{ [m]}, & R &= 0.09 \text{ [pixel}^2]. \end{aligned} \quad (4.5)$$

As discussed above, the true state, x_{true} , and (noise-corrupted) measurement, y_{meas} , are drawn randomly from $p(x)$ and $p(y|x)$, respectively. Each time we repeat the experiment, these values will change. In order to plot the posterior for a single experiment, we used the particular values

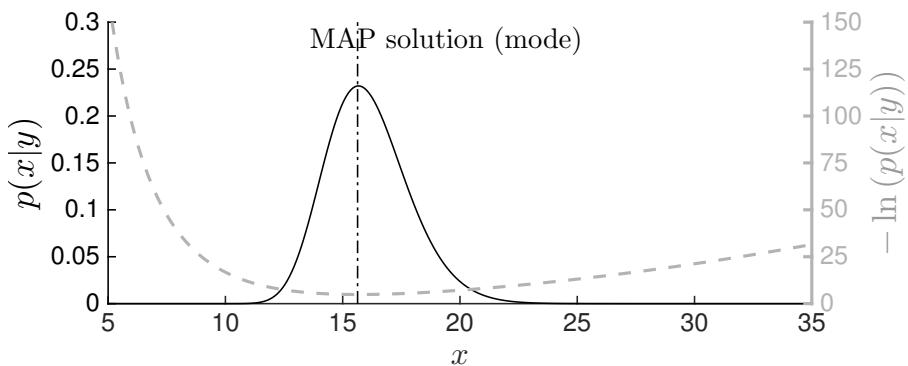
$$x_{\text{true}} = 22 \text{ [m]}, \quad y_{\text{meas}} = \frac{fb}{x_{\text{true}}} + 1 \text{ [pixel]},$$

which are fairly typical given the noise characteristics.

Figure 4.2 plots the prior and posterior for this example. Since we are considering a one-dimensional scenario, the denominator integral in (4.2) was computed numerically, and thus we effectively have a view of the full Bayesian posterior with no approximation. We can observe that even though the prior and measurement densities are Gaussian, the posterior is asymmetrical; it is skewed to one side by the nonlinear observation model. However, since the posterior is still unimodal (a single peak), we might still be justified in approximating it as Gaussian. This idea is discussed later in the chapter. We also see that the incorporation of the measurement results in a posterior that is more concentrated (i.e., more ‘certain’) about the state than the prior; this is the main idea behind Bayesian state estimation: *we want to incorporate measurements into the prior to become more certain about the posterior state*.

Unfortunately, while we were able to effectively compute the exact Bayesian posterior in our simple stereo camera example, this is typically

Figure 4.3
Posterior from stereo camera example, $p(x|y)$, as well as the negative log likelihood of the posterior, $-\ln(p(x|y))$ (dashed). We see that the MAP solution is simply the value of x that maximizes (or minimizes) either of these functions. In other words, the MAP solution is the *mode* of the posterior, which is not generally the same as the *mean*.



not tractable for real problems. As a result, a variety of tactics have been built up over the years to compute an approximate posterior. For example, the MAP approach is concerned with finding only the most likely state, or in other words the *mode* or ‘peak’ of the posterior. We discuss this next.

4.1.2 Maximum a Posteriori Estimation

As mentioned above, computing the full Bayesian posterior can be intractable in general. A very common approach is to seek out only the value of the state that maximizes the true posterior. This is called *maximum a posteriori (MAP)* estimation and is depicted graphically in Figure 4.3.

In other words, we want to compute

$$\hat{x}_{\text{map}} = \arg \max_x p(x|y). \quad (4.6)$$

Equivalently, we can try minimizing the negative log likelihood:

$$\hat{x}_{\text{map}} = \arg \min_x (-\ln(p(x|y))), \quad (4.7)$$

which can be easier when the PDFs involved are from the exponential family. As we are seeking only the most likely state, we can use Bayes’ rule to write

$$\hat{x}_{\text{map}} = \arg \min_x (-\ln(p(y|x)) - \ln(p(x))), \quad (4.8)$$

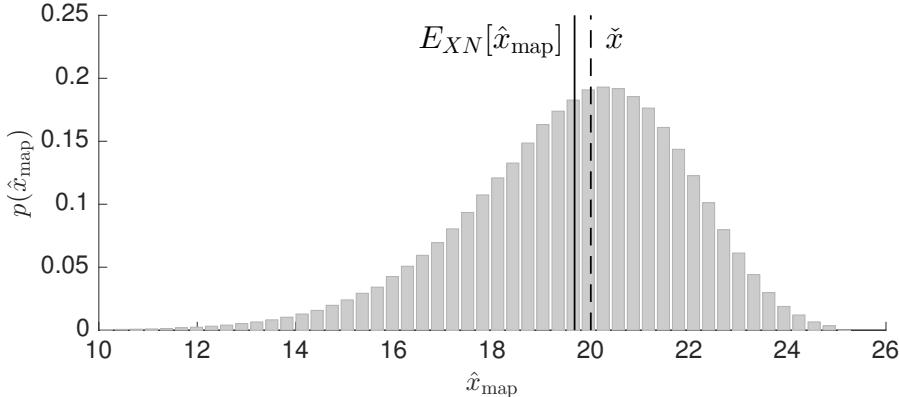
which drops $p(y)$ since it does not depend on x .

Relating this back to the stereo camera example presented earlier, we can write

$$\hat{x}_{\text{map}} = \arg \min_x J(x), \quad (4.9)$$

with

$$J(x) = \frac{1}{2R} \left(y - \frac{fb}{x} \right)^2 + \frac{1}{2\bar{P}} (\check{x} - x)^2, \quad (4.10)$$



where we have dropped any further normalization constants that do not depend on x . We can then find \hat{x}_{map} using any number of numerical optimization techniques.

Since the MAP estimator, \hat{x}_{map} , finds the most likely state given the data and prior, a question we might ask is, *how well does this estimator actually capture x_{true} ?* In robotics, we often report the average performance of our estimators, \hat{x} with respect to some ‘ground truth’. In other words, we compute

$$e_{\text{mean}}(\hat{x}) = E_{XN}[\hat{x} - x_{\text{true}}], \quad (4.11)$$

where $E_{XN}[\cdot]$ is the *expectation operator*; we explicitly include the subscripts XN to indicate that we are averaging over both the random draw of x_{true} from the prior and the random draw of n from the measurement noise. Since x_{true} is assumed to be independent of n , we have $E_{XN}[x_{\text{true}}] = E_X[x_{\text{true}}] = \check{x}$, and so

$$e_{\text{mean}}(\hat{x}) = E_{XN}[\hat{x}] - \check{x}. \quad (4.12)$$

It may be surprising to learn that under this performance measure, MAP estimation is *biased* (i.e., $e_{\text{mean}}(\hat{x}_{\text{map}}) \neq 0$). This can be attributed to the presence of a nonlinear measurement model, $g(\cdot)$, and the fact that the mode and mean of the posterior PDF are not the same. As discussed in the last chapter, when $g(\cdot)$ is linear, then $e_{\text{mean}}(\hat{x}_{\text{map}}) = 0$.

However, since we can trivially set the estimate to the prior, $\hat{x} = \check{x}$, and obtain $e_{\text{mean}}(\check{x}) = 0$, we need to define a secondary performance metric. This metric is typically the average squared error, e_{sq} , where

$$e_{\text{sq}}(\hat{x}) = E_{XN}[(\hat{x} - x_{\text{true}})^2]. \quad (4.13)$$

In other words, the first metric, e_{mean} , captures the mean of the estimator error, while the second, e_{sq} , captures the combined effects of bias and variance. Performing well on these two metrics results in the bias-variance tradeoff in the machine learning literature (Bishop, 2006).

Figure 4.4

Histogram of estimator values for 1,000,000 trials of the stereo camera experiment where each time a new x_{true} is randomly drawn from the prior and a new y_{meas} is randomly drawn from the measurement model. The dashed line marks the mean of the prior, \check{x} , and the solid line marks the expected value of the MAP estimator, \hat{x}_{map} , over all the trials. The gap between dashed and solid is $e_{\text{mean}} \approx -33.0 \text{ cm}$, which indicates a bias. The average squared error is $e_{\text{sq}} \approx 4.41 \text{ m}^2$.

Good performance on both metrics is necessary for a practical state estimator.

Figure 4.4 shows the MAP bias for the stereo camera example. We see that over a large number of trials (using the parameters in (4.5)), the average difference between the estimator, \hat{x}_{map} , and the ground-truth, $x_{\text{true}} = 20$ m, is $e_{\text{mean}} \approx -33.0$ cm, demonstrating a small bias. The average squared error is $e_{\text{sq}} \approx 4.41$ m².

Note that in this experiment we have drawn the true state from the prior used in the estimator, and we still see a bias. The bias can be worse in practice, as we often do not really know from which prior the true state is drawn and must invent one.

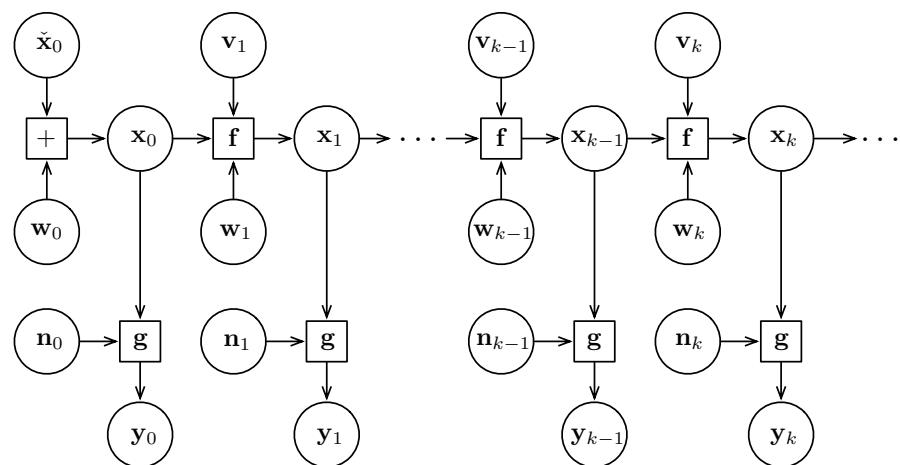
In the rest of this chapter, we will be discussing various estimation approaches for nonlinear, non-Gaussian systems. We must be careful to try to understand what aspect of the full Bayesian posterior each method captures: mean, mode, something else? We prefer to make distinctions in these terms rather than saying one method is more accurate than another. Accuracy can only really be compared fairly if two methods are trying to get to the same answer.

4.2 Recursive Discrete-Time Estimation

4.2.1 Problem Setup

Just as in the chapter on linear-Gaussian estimation, we require a set of motion and observation models upon which to base our estimator. We will consider discrete-time, time-invariant equations, but this time we will allow nonlinear equations (we will return to continuous time at the end of this chapter). We define the following motion and observation

Figure 4.5
Graphical model representation of the Markov process constituting the NLNG system described by (4.14).



models:

$$\text{motion model: } \mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{v}_k, \mathbf{w}_k), \quad k = 1 \dots K \quad (4.14\text{a})$$

$$\text{observation model: } \mathbf{y}_k = \mathbf{g}(\mathbf{x}_k, \mathbf{n}_k), \quad k = 0 \dots K \quad (4.14\text{b})$$

where k is again the discrete-time index and K its maximum. The function $\mathbf{f}(\cdot)$ is the nonlinear motion model and the function $\mathbf{g}(\cdot)$ is the nonlinear observation model. The variables take on the same meanings as in the linear-Gaussian chapter. For now we do not make any assumption about any of the random variables being Gaussian.

Figure 4.5 provides a graphical representation of the temporal evolution of the system described by (4.14). From this picture we can observe a very important characteristic of the system, the *Markov property*:

In the simplest sense, a stochastic process has the Markov property if the conditional probability density functions (PDFs) of future states of the process, given the present state, depend only upon the present state, but not on any other past states, i.e., they are conditionally independent of these older states. Such a process is called Markovian or a Markov process.

Our system is such a Markov process. For example, once we know the value of \mathbf{x}_{k-1} , we do not need to know the value of any previous states to evolve the system forward in time to compute \mathbf{x}_k . This property was exploited fully in the section on linear-Gaussian estimation. There it was assumed that we could employ this property in our estimator design, and this led to an elegant recursive estimator, the Kalman filter. But what about NLNG systems? Can we still have a recursive solution? The answer is yes, but only approximately. The next few sections will examine this claim.

4.2.2 Bayes Filter

In the chapter on linear-Gaussian estimation, we started with a batch estimation technique and worked down to the recursive Kalman filter. In this section, we will start by deriving a recursive filter, the Bayes filter (Jazwinski, 1970), and return to batch methods near the end of the chapter. This order reflects the historical sequence of events in the estimation world and will allow us to highlight exactly where the limiting assumptions and approximations have been made.

The Bayes filter seeks to come up with an entire PDF to represent the likelihood of the state, \mathbf{x}_k , using only measurements up to and including the current time. Using our notation from before, we want to compute

$$p(\mathbf{x}_k | \check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k}), \quad (4.15)$$

which is also sometimes called the *belief* for \mathbf{x}_k . Recall from the section

on factoring the batch linear-Gaussian solution that

$$p(\mathbf{x}_k | \mathbf{v}, \mathbf{y}) = \eta \underbrace{p(\mathbf{x}_k | \check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k})}_{\text{forwards}} \underbrace{p(\mathbf{x}_k | \mathbf{v}_{k+1:K}, \mathbf{y}_{k+1:K})}_{\text{backwards}}. \quad (4.16)$$

Thus, in this section, we will focus on turning the ‘forwards’ PDF into a recursive filter (for nonlinear non-Gaussian systems). By employing the independence of all the measurements², we may factor out the latest measurement to have

$$p(\mathbf{x}_k | \check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k}) = \eta p(\mathbf{y}_k | \mathbf{x}_k) p(\mathbf{x}_k | \check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k-1}), \quad (4.17)$$

where we have employed Bayes’ rule to reverse the dependence and η serves to preserve the axiom of total probability. Turning our attention to the second factor, we introduce the hidden state, \mathbf{x}_{k-1} , and integrate over all possible values:

$$\begin{aligned} p(\mathbf{x}_k | \check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k-1}) &= \int p(\mathbf{x}_k, \mathbf{x}_{k-1} | \check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k-1}) d\mathbf{x}_{k-1} \\ &= \int p(\mathbf{x}_k | \mathbf{x}_{k-1}, \check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k-1}) p(\mathbf{x}_{k-1} | \check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k-1}) d\mathbf{x}_{k-1}. \end{aligned} \quad (4.18)$$

The introduction of the hidden state can be viewed as the opposite of marginalization. So far we have not introduced any approximations. The next step is subtle and is the cause of many limitations in recursive estimation. Since our system enjoys the Markov property, we use said property (on the estimator) to say that

$$p(\mathbf{x}_k | \mathbf{x}_{k-1}, \check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k-1}) = p(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{v}_k), \quad (4.19a)$$

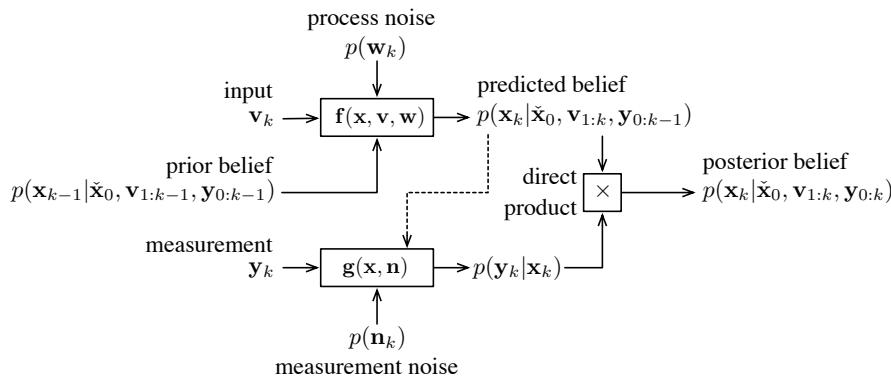
$$p(\mathbf{x}_{k-1} | \check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k-1}) = p(\mathbf{x}_{k-1} | \check{\mathbf{x}}_0, \mathbf{v}_{1:k-1}, \mathbf{y}_{0:k-1}), \quad (4.19b)$$

which seems entirely reasonable given the depiction in Figure 4.5. However, we will come back to examine (4.19) later in this chapter. Substituting (4.19) and (4.18) into (4.17), we have the Bayes filter³:

$$\begin{aligned} &\underbrace{p(\mathbf{x}_k | \check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k})}_{\text{posterior belief}} \\ &= \eta \underbrace{p(\mathbf{y}_k | \mathbf{x}_k)}_{\substack{\text{observation} \\ \text{correction} \\ \text{using } \mathbf{g}(\cdot)}} \underbrace{\int p(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{v}_k) p(\mathbf{x}_{k-1} | \check{\mathbf{x}}_0, \mathbf{v}_{1:k-1}, \mathbf{y}_{0:k-1}) d\mathbf{x}_{k-1}}_{\substack{\text{motion} \\ \text{prediction} \\ \text{using } \mathbf{f}(\cdot)}} \underbrace{p(\mathbf{x}_{k-1} | \check{\mathbf{x}}_0, \mathbf{v}_{1:k-1}, \mathbf{y}_{0:k-1})}_{\text{prior belief}} \end{aligned} \quad (4.20)$$

² We will continue to assume all the measurements are statistically independent as in the LG case.

³ There is a special case at the first timestep, $k = 0$, that only involves the observation correction with \mathbf{y}_0 , but we omit it to avoid complexity. We assume the filter is initialized with $p(\mathbf{x}_0 | \check{\mathbf{x}}_0, \mathbf{y}_0)$.



We can see that (4.20) takes on a predictor-corrector form. In the prediction step, the prior⁴ belief, $p(\mathbf{x}_{k-1}|\tilde{\mathbf{x}}_0, \mathbf{v}_{1:k-1}, \mathbf{y}_{0:k-1})$, is propagated forward in time using the input, \mathbf{v}_k , and the motion model, $\mathbf{f}(\cdot)$. In the correction step, the predicted estimate is then updated using the measurement, \mathbf{y}_k , and the measurement model, $\mathbf{g}(\cdot)$. The result is the posterior belief, $p(\mathbf{x}_k|\tilde{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k})$. Figure 4.6 provides a graphical depiction of the information flow in the Bayes filter. The important message to take away from these diagrams is that we require methods of passing PDFs through the nonlinear functions, $\mathbf{f}(\cdot)$ and $\mathbf{g}(\cdot)$.

Although exact, the Bayes filter is really nothing more than a mathematical artifact; it can never be implemented in practice, except for the linear-Gaussian case. There are two primary reasons for this, and as such we need to make appropriate approximations:

- (i) Probability density functions live in an infinite-dimensional space (as do all continuous functions) and as such an infinite amount of memory (i.e., infinite number of parameters) would be needed to completely represent the belief, $p(\mathbf{x}_k|\tilde{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k})$. To overcome this memory issue, the belief is approximately represented. One approach is to approximate this function as a Gaussian PDF (i.e., keep track of the first two moments, mean and covariance). Another approach is to approximate the PDF using a finite number of random samples. We will look into both of these later on.
- (ii) The integral in the Bayes filter is computationally very expensive; it would require infinite computing resources to evaluate exactly. To overcome this computational resource issue, the integral must be evaluated approximately. One approach is to linearize the motion and observation models and then eval-

Figure 4.6
Graphical depictions of the Bayes filter. Here the dashed line indicates that in practice a hint could be passed to the ‘correction step’ about the states in which the probability mass of the belief function is concentrated. This can be used to reduce the need to work out the full density for the observation, \mathbf{y}_k , given the state, \mathbf{x}_k .

⁴ To be clear, the Bayes filter is using Bayesian inference, but just at a single timestep. The batch methods discussed in the previous chapter performed inference over the whole trajectory at once. We will return to the batch situation later in this chapter.

ate the integrals in closed form. Another approach is to employ *Monte Carlo integration*. We will look into both of these later on as well.

Much of the research in recursive state estimation has focused on better and better approximations to handle these two issues. Considerable gains have been made that are worth examining in more detail. As such, we will look at some of the classic and modern approaches to approximate the Bayes filter in the next few sections. However, we must keep in our minds the assumption on which the Bayes filter is predicated: the Markov property. A question we must ask ourselves is, what happens to the Markov property when we start making these approximations to the Bayes filter? We will return to this later. For now, let us assume it holds.

4.2.3 Extended Kalman Filter

We now show that if the belief is constrained to be Gaussian, the noise is Gaussian, and we linearize the motion and observation models in order to carry out the integral (and also the normalized product) in the Bayes filter, we arrive at the famous *extended Kalman filter (EKF)*⁵. The EKF is still the mainstay of estimation and data fusion in many circles, and can often be effective for mildly nonlinear, non-Gaussian systems. For a good reference on the EKF, see Maybeck (1994).

The EKF was a key tool used to estimate spacecraft trajectories on the NASA Apollo program. Shortly after Kalman's original paper (Kalman, 1960b) was published, he met with Stanley F. Schmidt of NASA Ames Research Center. Schmidt was impressed with Kalman's filter and his team went on to modify it to work for their task of spacecraft navigation. In particular, they (i) extended it to work for nonlinear motion and observation models, (ii) came up with the idea of linearizing about the best current estimate to reduce nonlinear effects, and (iii) reformulated the original filter to the now-standard separate prediction and correction steps (McGee and Schmidt, 1985). For these significant contributions, the EKF was sometimes called the Schmidt-Kalman filter, but this name has fallen out of favour due to confusion with another similarly named contribution later made by Schmidt (to account for unobservable biases while keeping state dimension low). Schmidt also went on to work on the square-root formulation of the EKF to improve numerical stability (Bierman, 1974). Later, at Lockheed Missiles and Space Company, Schmidt's popularization of Kalman's work also inspired Charlotte Striebel to begin work on connecting the KF

Stanley F. Schmidt (1926-) is an American aerospace engineer who adapted Kalman's filter early on to estimate spacecraft trajectories on the Apollo program. It was his work that led to what is now called the extended Kalman filter.

⁵ The EKF is called ‘extended’ because it is the extension of the Kalman filter to nonlinear systems.

to other types of trajectory estimation, which ultimately led to the Rauch-Tung-Striebel smoother discussed in the previous chapter.

To derive the EKF, we first limit (i.e., constrain) our belief function for \mathbf{x}_k to be Gaussian:

$$p(\mathbf{x}_k | \check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k}) = \mathcal{N}(\hat{\mathbf{x}}_k, \hat{\mathbf{P}}_k), \quad (4.21)$$

where $\hat{\mathbf{x}}_k$ is the mean and $\hat{\mathbf{P}}_k$ the covariance. Next, we assume that the noise variables, \mathbf{w}_k and \mathbf{n}_k ($\forall k$), are in fact Gaussian as well:

$$\mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_k), \quad (4.22a)$$

$$\mathbf{n}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k). \quad (4.22b)$$

Note that a Gaussian PDF can be transformed through a nonlinearity to be non-Gaussian. In fact, we will look at this in more detail a bit later in this chapter. We assume this is the case for the noise variables; in other words, the nonlinear motion and observation models may affect \mathbf{w}_k and \mathbf{n}_k . They are not necessarily added after the nonlinearities, as in

$$\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{v}_k) + \mathbf{w}_k, \quad (4.23a)$$

$$\mathbf{y}_k = \mathbf{g}(\mathbf{x}_k) + \mathbf{n}_k, \quad (4.23b)$$

but rather appear inside the nonlinearities as in (4.14). Equations (4.23) are in fact a special case of (4.14). However, we can recover additive noise (approximately) through linearization, which we show next.

With $\mathbf{g}(\cdot)$ and $\mathbf{f}(\cdot)$ nonlinear, we still cannot compute the integral in the Bayes filter in closed form, so we turn to linearization. We linearize the motion and observation models about the current state estimate mean:

$$\mathbf{f}(\mathbf{x}_{k-1}, \mathbf{v}_k, \mathbf{w}_k) \approx \check{\mathbf{x}}_k + \mathbf{F}_{k-1}(\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1}) + \mathbf{w}'_k, \quad (4.24a)$$

$$\mathbf{g}(\mathbf{x}_k, \mathbf{n}_k) \approx \check{\mathbf{y}}_k + \mathbf{G}_k(\mathbf{x}_k - \check{\mathbf{x}}_k) + \mathbf{n}'_k, \quad (4.24b)$$

where

$$\check{\mathbf{x}}_k = \mathbf{f}(\hat{\mathbf{x}}_{k-1}, \mathbf{v}_k, \mathbf{0}), \quad \mathbf{F}_{k-1} = \frac{\partial \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{v}_k, \mathbf{w}_k)}{\partial \mathbf{x}_{k-1}} \Big|_{\hat{\mathbf{x}}_{k-1}, \mathbf{v}_k, \mathbf{0}}, \quad (4.25a)$$

$$\mathbf{w}'_k = \frac{\partial \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{v}_k, \mathbf{w}_k)}{\partial \mathbf{w}_k} \Big|_{\hat{\mathbf{x}}_{k-1}, \mathbf{v}_k, \mathbf{0}} \mathbf{w}_k, \quad (4.25b)$$

and

$$\check{\mathbf{y}}_k = \mathbf{g}(\check{\mathbf{x}}_k, \mathbf{0}), \quad \mathbf{G}_k = \frac{\partial \mathbf{g}(\mathbf{x}_k, \mathbf{n}_k)}{\partial \mathbf{x}_k} \Big|_{\check{\mathbf{x}}_k, \mathbf{0}}, \quad (4.26a)$$

$$\mathbf{n}'_k = \frac{\partial \mathbf{g}(\mathbf{x}_k, \mathbf{n}_k)}{\partial \mathbf{n}_k} \Big|_{\check{\mathbf{x}}_k, \mathbf{0}} \mathbf{n}_k. \quad (4.26b)$$

From here the statistical properties of the current state, \mathbf{x}_k , given the old state and latest input, are

$$\mathbf{x}_k \approx \check{\mathbf{x}}_k + \mathbf{F}_{k-1} (\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1}) + \mathbf{w}'_k, \quad (4.27a)$$

$$E[\mathbf{x}_k] \approx \check{\mathbf{x}}_k + \mathbf{F}_{k-1} (\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1}) + \underbrace{E[\mathbf{w}'_k]}_0, \quad (4.27b)$$

$$E[(\mathbf{x}_k - E[\mathbf{x}_k])(\mathbf{x}_k - E[\mathbf{x}_k])^T] \approx \underbrace{E[\mathbf{w}'_k \mathbf{w}'_k^T]}_{\mathbf{Q}'_k}, \quad (4.27c)$$

$$p(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{v}_k) \approx \mathcal{N}(\check{\mathbf{x}}_k + \mathbf{F}_{k-1} (\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1}), \mathbf{Q}'_k). \quad (4.27d)$$

For the statistical properties of the current measurement, \mathbf{y}_k , given the current state, we have

$$\mathbf{y}_k \approx \check{\mathbf{y}}_k + \mathbf{G}_k (\mathbf{x}_k - \check{\mathbf{x}}_k) + \mathbf{n}'_k, \quad (4.28a)$$

$$E[\mathbf{y}_k] \approx \check{\mathbf{y}}_k + \mathbf{G}_k (\mathbf{x}_k - \check{\mathbf{x}}_k) + \underbrace{E[\mathbf{n}'_k]}_0, \quad (4.28b)$$

$$E[(\mathbf{y}_k - E[\mathbf{y}_k])(\mathbf{y}_k - E[\mathbf{y}_k])^T] \approx \underbrace{E[\mathbf{n}'_k \mathbf{n}'_k^T]}_{\mathbf{R}'_k}, \quad (4.28c)$$

$$p(\mathbf{y}_k | \mathbf{x}_k) \approx \mathcal{N}(\check{\mathbf{y}}_k + \mathbf{G}_k (\mathbf{x}_k - \check{\mathbf{x}}_k), \mathbf{R}'_k). \quad (4.28d)$$

Substituting in these results, the Bayes filter becomes

$$\begin{aligned} & \underbrace{p(\mathbf{x}_k | \check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k})}_{\mathcal{N}(\hat{\mathbf{x}}_k, \hat{\mathbf{P}}_k)} = \eta \underbrace{p(\mathbf{y}_k | \mathbf{x}_k)}_{\mathcal{N}(\check{\mathbf{y}}_k + \mathbf{G}_k (\mathbf{x}_k - \check{\mathbf{x}}_k), \mathbf{R}'_k)} \\ & \times \int \underbrace{p(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{v}_k)}_{\mathcal{N}(\check{\mathbf{x}}_k + \mathbf{F}_{k-1} (\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1}), \mathbf{Q}'_k)} \underbrace{p(\mathbf{x}_{k-1} | \check{\mathbf{x}}_0, \mathbf{v}_{1:k-1}, \mathbf{y}_{0:k-1})}_{\mathcal{N}(\hat{\mathbf{x}}_{k-1}, \hat{\mathbf{P}}_{k-1})} d\mathbf{x}_{k-1}. \end{aligned} \quad (4.29)$$

Using our formula (2.90) for passing a Gaussian through a (stochastic) nonlinearity, we can see that the integral is also Gaussian:

$$\begin{aligned} & \underbrace{p(\mathbf{x}_k | \check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k})}_{\mathcal{N}(\hat{\mathbf{x}}_k, \hat{\mathbf{P}}_k)} = \eta \underbrace{p(\mathbf{y}_k | \mathbf{x}_k)}_{\mathcal{N}(\check{\mathbf{y}}_k + \mathbf{G}_k (\mathbf{x}_k - \check{\mathbf{x}}_k), \mathbf{R}'_k)} \\ & \times \underbrace{\int p(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{v}_k) p(\mathbf{x}_{k-1} | \check{\mathbf{x}}_0, \mathbf{v}_{1:k-1}, \mathbf{y}_{0:k-1}) d\mathbf{x}_{k-1}}_{\mathcal{N}(\check{\mathbf{x}}_k + \mathbf{F}_{k-1} \hat{\mathbf{P}}_{k-1} \mathbf{F}_{k-1}^T + \mathbf{Q}'_k)}. \end{aligned} \quad (4.30)$$

We are now left with the normalized product of two Gaussian PDFs, which we also discussed previously in Section 2.2.6. Applying (2.70),

we find that

$$\begin{aligned} & \underbrace{p(\mathbf{x}_k | \check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k})}_{\mathcal{N}(\hat{\mathbf{x}}_k, \hat{\mathbf{P}}_k)} \\ &= \eta p(\mathbf{y}_k | \mathbf{x}_k) \underbrace{\int p(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{v}_k) p(\mathbf{x}_{k-1} | \check{\mathbf{x}}_0, \mathbf{v}_{1:k-1}, \mathbf{y}_{0:k-1}) d\mathbf{x}_{k-1}}_{\mathcal{N}(\check{\mathbf{x}}_k + \mathbf{K}_k(\mathbf{y}_k - \check{\mathbf{y}}_k), (\mathbf{I} - \mathbf{K}_k \mathbf{G}_k)(\mathbf{F}_{k-1} \hat{\mathbf{P}}_{k-1} \mathbf{F}_{k-1}^T + \mathbf{Q}'_k))}, \quad (4.31) \end{aligned}$$

where \mathbf{K}_k is known as the Kalman gain matrix (given below). Getting to this last line takes quite a bit of tedious algebra and is left to the reader. Comparing left and right sides of our posterior expression above, we have

$$\begin{aligned} \text{predictor: } & \check{\mathbf{P}}_k = \mathbf{F}_{k-1} \hat{\mathbf{P}}_{k-1} \mathbf{F}_{k-1}^T + \mathbf{Q}'_k, & (4.32a) \\ & \check{\mathbf{x}}_k = \mathbf{f}(\hat{\mathbf{x}}_{k-1}, \mathbf{v}_k, \mathbf{0}), & (4.32b) \\ \text{Kalman gain: } & \mathbf{K}_k = \check{\mathbf{P}}_k \mathbf{G}_k^T (\mathbf{G}_k \check{\mathbf{P}}_k \mathbf{G}_k^T + \mathbf{R}'_k)^{-1}, & (4.32c) \\ \text{corrector: } & \hat{\mathbf{P}}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{G}_k) \check{\mathbf{P}}_k, & (4.32d) \\ & \hat{\mathbf{x}}_k = \check{\mathbf{x}}_k + \mathbf{K}_k \underbrace{(\mathbf{y}_k - \mathbf{g}(\check{\mathbf{x}}_k, \mathbf{0}))}_{\text{innovation}}. & (4.32e) \end{aligned}$$

Equations (4.32) are known as the classic recursive update equations for the EKF. The update equations allow us to compute $\{\hat{\mathbf{x}}_k, \hat{\mathbf{P}}_k\}$ from $\{\hat{\mathbf{x}}_{k-1}, \hat{\mathbf{P}}_{k-1}\}$. We notice immediately the similar structure to (3.120) for linear-Gaussian estimation. There are two main differences here:

- (i) The nonlinear motion and observation models are used to propagate the mean of our estimate.
- (ii) There are Jacobians embedded in the \mathbf{Q}'_k and \mathbf{R}'_k covariances for the noise. This comes from the fact that we allowed the noise to be applied within the nonlinearities in (4.14).

It should be noted that there is no guarantee that the EKF will perform adequately for a general nonlinear system. To gauge the performance of the EKF on a particular nonlinear system, it often comes down to simply trying it out. The main problem with the EKF is the operating point of the linearization is the mean of our estimate of the state, not the true state. This seemingly small difference can cause the EKF to diverge wildly in some cases. Sometimes the result is less dramatic, with the estimate being *biased* or *inconsistent* or, most often, both.

4.2.4 Generalized Gaussian Filter

The Bayes filter is appealing because it can be written out exactly. We can then reach a number of implementable filters through different

approximations on the form of the estimated PDF and handling methods. There is, however, a cleaner approach to deriving those filters that assume up front that the estimated PDF is Gaussian. We have actually already seen this in practice in Section 3.3.3, where we derived the Kalman filter using Bayesian inference.

In general, we begin with a Gaussian prior at time $k - 1$:

$$p(\mathbf{x}_{k-1} | \check{\mathbf{x}}_0, \mathbf{v}_{1:k-1}, \mathbf{y}_{0:k-1}) = \mathcal{N}(\hat{\mathbf{x}}_{k-1}, \hat{\mathbf{P}}_{k-1}). \quad (4.33)$$

We pass this forwards in time through the nonlinear motion model, $\mathbf{f}(\cdot)$, to propose a Gaussian prior at time k :

$$p(\mathbf{x}_k | \check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k-1}) = \mathcal{N}(\check{\mathbf{x}}_k, \check{\mathbf{P}}_k). \quad (4.34)$$

This is the *prediction step* and incorporates the latest input, \mathbf{v}_k .

For the *correction step*, we employ the method from Section 2.2.3 and write a joint Gaussian for the state and latest measurement, at time k :

$$p(\mathbf{x}_k, \mathbf{y}_k | \check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k-1}) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_{x,k} \\ \boldsymbol{\mu}_{y,k} \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{xx,k} & \boldsymbol{\Sigma}_{xy,k} \\ \boldsymbol{\Sigma}_{yx,k} & \boldsymbol{\Sigma}_{yy,k} \end{bmatrix}\right). \quad (4.35)$$

We then write the conditional Gaussian density for \mathbf{x}_k (i.e., the posterior) directly as

$$\begin{aligned} p(\mathbf{x}_k | \check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k}) \\ = \mathcal{N}\left(\underbrace{\boldsymbol{\mu}_{x,k} + \boldsymbol{\Sigma}_{xy,k} \boldsymbol{\Sigma}_{yy,k}^{-1} (\mathbf{y}_k - \boldsymbol{\mu}_{y,k})}_{\hat{\mathbf{x}}_k}, \underbrace{\boldsymbol{\Sigma}_{xx,k} - \boldsymbol{\Sigma}_{xy,k} \boldsymbol{\Sigma}_{yy,k}^{-1} \boldsymbol{\Sigma}_{yx,k}}_{\hat{\mathbf{P}}_k}\right), \end{aligned} \quad (4.36)$$

where we have defined $\hat{\mathbf{x}}_k$ as the mean and $\hat{\mathbf{P}}_k$ as the covariance. The nonlinear observation model, $\mathbf{g}(\cdot)$, is used in the computation of $\boldsymbol{\mu}_{y,k}$. From here, we can write the generalized Gaussian correction-step equations as

$$\mathbf{K}_k = \boldsymbol{\Sigma}_{xy,k} \boldsymbol{\Sigma}_{yy,k}^{-1}, \quad (4.37a)$$

$$\hat{\mathbf{P}}_k = \check{\mathbf{P}}_k - \mathbf{K}_k \boldsymbol{\Sigma}_{xy,k}^T, \quad (4.37b)$$

$$\hat{\mathbf{x}}_k = \check{\mathbf{x}}_k + \mathbf{K}_k (\mathbf{y}_k - \boldsymbol{\mu}_{y,k}), \quad (4.37c)$$

where we have let $\boldsymbol{\mu}_{x,k} = \check{\mathbf{x}}_k$, $\boldsymbol{\Sigma}_{xx,k} = \check{\mathbf{P}}_k$, and \mathbf{K}_k is still known as the Kalman gain. Unfortunately, unless the motion and observation models are linear, we cannot compute all the remaining quantities required exactly: $\boldsymbol{\mu}_{y,k}$, $\boldsymbol{\Sigma}_{yy,k}$, and $\boldsymbol{\Sigma}_{xy,k}$. This is because putting a Gaussian PDF into a nonlinearity generally results in something non-Gaussian coming out the other end. We therefore need to consider approximations at this stage.

The next section will revisit linearizing the motion and observation models to complete this cleaner derivation of the EKF. After that,

we will discuss other methods of passing PDFs through nonlinearities, which lead to other flavours of the Bayes and Kalman filters.

4.2.5 Iterated Extended Kalman Filter

Continuing on from the last section, we complete our alternate derivation of the *iterated extended Kalman filter (IEKF)*. The *prediction step* is fairly straightforward and essentially the same as Section 4.2.3. We therefore omit it but note that the prior, at time k , is

$$p(\mathbf{x}_k | \check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k-1}) = \mathcal{N}(\check{\mathbf{x}}_k, \check{\mathbf{P}}_k), \quad (4.38)$$

which incorporates \mathbf{v}_k .

The *correction step* is where things become a little more interesting. Our *nonlinear* measurement model is given by

$$\mathbf{y}_k = \mathbf{g}(\mathbf{x}_k, \mathbf{n}_k). \quad (4.39)$$

We linearize about an arbitrary operating point, $\mathbf{x}_{\text{op},k}$:

$$\mathbf{g}(\mathbf{x}_k, \mathbf{n}_k) \approx \mathbf{y}_{\text{op},k} + \mathbf{G}_k (\mathbf{x}_k - \mathbf{x}_{\text{op},k}) + \mathbf{n}'_k, \quad (4.40)$$

where

$$\mathbf{y}_{\text{op},k} = \mathbf{g}(\mathbf{x}_{\text{op},k}, \mathbf{0}), \quad \mathbf{G}_k = \left. \frac{\partial \mathbf{g}(\mathbf{x}_k, \mathbf{n}_k)}{\partial \mathbf{x}_k} \right|_{\mathbf{x}_{\text{op},k}, \mathbf{0}}, \quad (4.41a)$$

$$\mathbf{n}'_k = \left. \frac{\partial \mathbf{g}(\mathbf{x}_k, \mathbf{n}_k)}{\partial \mathbf{n}_k} \right|_{\mathbf{x}_{\text{op},k}, \mathbf{0}} \mathbf{n}_k. \quad (4.41b)$$

Note that the observation model and Jacobians are evaluated at $\mathbf{x}_{\text{op},k}$.

Using this linearized model, we can then express the joint density for the state and the measurement at time k as approximately Gaussian:

$$\begin{aligned} p(\mathbf{x}_k, \mathbf{y}_k | \check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k-1}) &\approx \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_{x,k} \\ \boldsymbol{\mu}_{y,k} \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{xx,k} & \boldsymbol{\Sigma}_{xy,k} \\ \boldsymbol{\Sigma}_{yx,k} & \boldsymbol{\Sigma}_{yy,k} \end{bmatrix}\right) \\ &= \mathcal{N}\left(\begin{bmatrix} \check{\mathbf{x}}_k \\ \mathbf{y}_{\text{op},k} + \mathbf{G}_k(\check{\mathbf{x}}_k - \mathbf{x}_{\text{op},k}) \end{bmatrix}, \begin{bmatrix} \check{\mathbf{P}}_k & \check{\mathbf{P}}_k \mathbf{G}_k^T \\ \mathbf{G}_k \check{\mathbf{P}}_k & \mathbf{G}_k \check{\mathbf{P}}_k \mathbf{G}_k^T + \mathbf{R}'_k \end{bmatrix}\right). \end{aligned} \quad (4.42)$$

Once again, if the measurement, \mathbf{y}_k , is known, we can use (2.53b) to write the Gaussian conditional density for \mathbf{x}_k (i.e., the posterior) as

$$\begin{aligned} p(\mathbf{x}_k | \check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k}) &= \mathcal{N}\left(\underbrace{\boldsymbol{\mu}_{x,k} + \boldsymbol{\Sigma}_{xy,k} \boldsymbol{\Sigma}_{yy,k}^{-1} (\mathbf{y}_k - \boldsymbol{\mu}_{y,k})}_{\hat{\mathbf{x}}_k}, \underbrace{\boldsymbol{\Sigma}_{xx,k} - \boldsymbol{\Sigma}_{xy,k} \boldsymbol{\Sigma}_{yy,k}^{-1} \boldsymbol{\Sigma}_{yx,k}}_{\hat{\mathbf{P}}_k}\right), \end{aligned} \quad (4.43)$$

where again we have defined $\hat{\mathbf{x}}_k$ as the mean and $\hat{\mathbf{P}}_k$ as the covariance.

As shown in the last section, the generalized Gaussian correction-step equations are

$$\mathbf{K}_k = \boldsymbol{\Sigma}_{xy,k} \boldsymbol{\Sigma}_{yy,k}^{-1}, \quad (4.44a)$$

$$\hat{\mathbf{P}}_k = \check{\mathbf{P}}_k - \mathbf{K}_k \boldsymbol{\Sigma}_{xy,k}^T, \quad (4.44b)$$

$$\hat{\mathbf{x}}_k = \check{\mathbf{x}}_k + \mathbf{K}_k (\mathbf{y}_k - \boldsymbol{\mu}_{y,k}). \quad (4.44c)$$

Substituting in the moments $\boldsymbol{\mu}_{y,k}$, $\boldsymbol{\Sigma}_{yy,k}$, and $\boldsymbol{\Sigma}_{xy,k}$ from above, we have

$$\mathbf{K}_k = \check{\mathbf{P}}_k \mathbf{G}_k^T (\mathbf{G}_k \check{\mathbf{P}}_k \mathbf{G}_k^T + \mathbf{R}'_k)^{-1}, \quad (4.45a)$$

$$\hat{\mathbf{P}}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{G}_k) \check{\mathbf{P}}_k, \quad (4.45b)$$

$$\hat{\mathbf{x}}_k = \check{\mathbf{x}}_k + \mathbf{K}_k (\mathbf{y}_k - \mathbf{y}_{op,k} - \mathbf{G}_k (\check{\mathbf{x}}_k - \mathbf{x}_{op,k})). \quad (4.45c)$$

These equations are very similar to the Kalman gain and corrector equations in (4.32); the only difference is the operating point of the linearization. If we set the operating point of the linearization to be the mean of the predicted prior, $\mathbf{x}_{op,k} = \check{\mathbf{x}}_k$, then (4.45) and (4.32) are identical.

However, it turns out that we can do much better if we iteratively recompute (4.45), each time setting the operating point to be the mean of the posterior at the last iteration:

$$\mathbf{x}_{op,k} \leftarrow \hat{\mathbf{x}}_k. \quad (4.46)$$

At the first iteration we take $\mathbf{x}_{op,k} = \check{\mathbf{x}}_k$. This allows us to be linearizing about better and better estimates, thereby improving our approximation each iteration. We terminate the process when the change to $\mathbf{x}_{op,k}$ from one iteration to the next is sufficiently small. Note that the covariance equation need only be computed once, after the other two equations converge.

4.2.6 IEKF Is a MAP Estimator

A great question to ask at this point is, *what is the relationship between the EKF/IEKF estimate and the full Bayesian posterior?* It turns out that the IEKF estimate corresponds to a (local) maximum of the full posterior⁶; in other words, it is a MAP estimate. On the other hand, since the EKF is not iterated, it can be very far from a local maximum; there is actually very little we can say about its relationship to the full posterior.

These relations are illustrated in Figure 4.7, where we compare the correction steps of the IEKF and EKF to the full Bayesian posterior on our stereo camera example introduced in Section 4.1.2. In this version

⁶ To be clear, this is only true for the correction step at a single timestep.

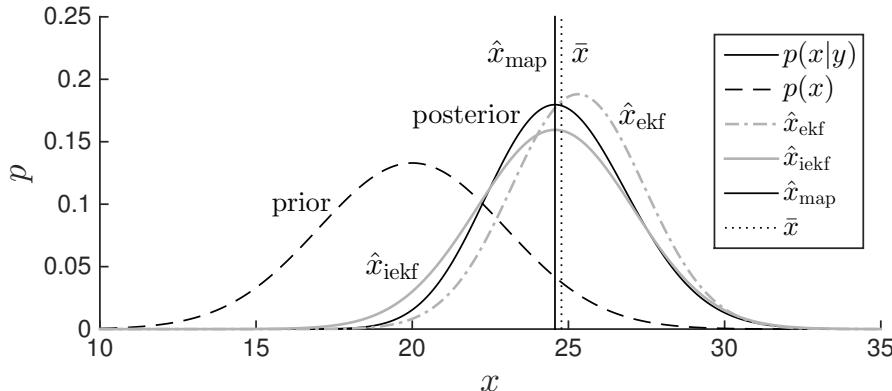


Figure 4.7 Stereo camera example, comparing the inference (i.e., ‘corrective’) step of the EKF and IEKF to the full Bayesian posterior, $p(x|y)$. We see that the mean of the IEKF matches up against the MAP solution, \hat{x}_{map} , while the EKF does not. The actual mean of the posterior is denoted \bar{x} .

of the example, we used

$$x_{\text{true}} = 26 \text{ [m]}, \quad y_{\text{meas}} = \frac{fb}{x_{\text{true}}} - 0.6 \text{ [pixel]},$$

to exaggerate the difference between the methods. As discussed above, the mean of the IEKF corresponds to the MAP (i.e., mode) solution, while the EKF is not easily relatable to the full posterior.

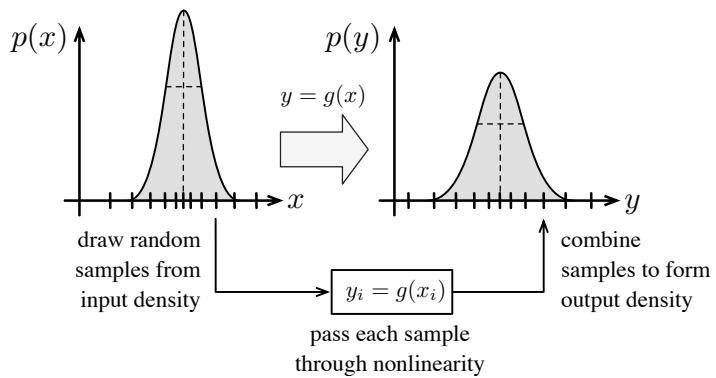
To understand why the IEKF is the same as the MAP estimate, we require some optimization tools that we will introduce later in the chapter. For now, the important take-away message from this section is that our choice to iteratively relinearize about our best guess leads to a MAP solution. Thus, the ‘mean’ of our IEKF Gaussian estimator does not actually match the mean of the full Bayesian posterior; it matches the mode.

4.2.7 Alternatives for Passing PDFs through Nonlinearities

In our derivation of the EKF/ IEKF, we used one particular technique to pass a PDF through a nonlinearity. Specifically, we linearized the nonlinear model about an operating point and then passed our Gaussian PDFs through the linearized model analytically. This is certainly one approach, but there are others. This section will discuss three common approaches: the Monte Carlo method (brute force), linearization (as in the EKF), and the *sigmapoint* or *unscented*⁷ transformation. Our motivation is to introduce some tools that can be used within our Bayes filter framework to derive alternatives to the EKF/ IEKF.

⁷ This name lives on in the literature; apparently, Simon Julier named it after an unscented deodorant to make the point that often we take names for granted without knowing their origins.

Figure 4.8 Monte Carlo method to transform a PDF through a nonlinearity. A large number of random samples are drawn from the input density, passed through the nonlinearity, and then used to form the output density.



Monte Carlo Method

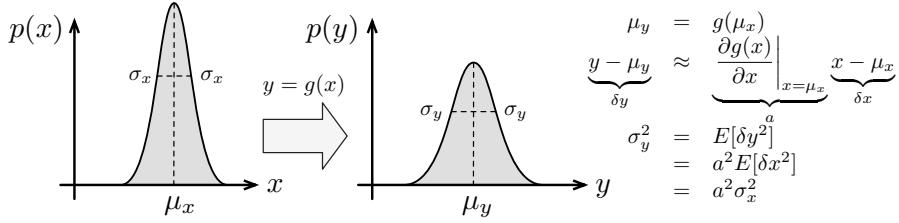
The Monte Carlo method of transforming a PDF through a nonlinearity is essentially the ‘brute force’ approach. The process is depicted in Figure 4.8. We draw a large number of samples from the input density, transform each one of these samples through the nonlinearity exactly, and then build the output density from the transformed samples (e.g., by computing the statistical moments). Loosely, the *law of large numbers* ensures this procedure will converge to the correct answer as the number of samples used approaches infinity.

The obvious problem with this method is that it can be terribly inefficient, particularly in higher dimensions. Aside from this obvious disadvantage, there are actually some advantages to this method:

- (i) It works with any PDF, not just Gaussian.
- (ii) It handles any type of nonlinearity (no requirement for differentiable or even continuous).
- (iii) We do not need to know the mathematical form of the nonlinear function – in practice the nonlinearity could be any software function.
- (iv) It is an ‘anytime’ algorithm – we can easily trade off accuracy against speed by choosing the number of samples appropriately.

Because we can make this method highly accurate, we can also use it to gauge the performance of other methods.

The other point worth mentioning at this stage is that the mean of the output density is not the same as the mean of the input density after being passed through the nonlinearity. This can be seen by way of a simple example. Consider the input density for x to be uniform over the interval $[0, 1]$; in other words, $p(x) = 1$, $x \in [0, 1]$. Let the nonlinearity be $y = x^2$. The mean of the input is $\mu_x = 1/2$, and passing this through the nonlinearity gives $\mu_y = 1/4$. However, the actual mean of the output is $\mu_y = \int_0^1 p(x) x^2 dx = 1/3$. Similar things happen to the higher statistical moments. The Monte Carlo method is able to



approach the correct answer with a large number of samples, but as we will see, some of the other methods cannot.

Linearization

The most popular method of transforming a Gaussian PDF through a nonlinearity is linearization, which we have already used to derive the EKF/ IEKF. Technically, the mean is actually passed through the nonlinearity exactly, while the covariance is approximately passed through a linearized version of the function. Typically, the operating point of the linearization process is the mean of the PDF. This procedure is depicted in Figure 4.9 (repeat of Figure 2.5 for convenience). This procedure is highly inaccurate for the following reasons:

- (i) The outcome of passing a Gaussian PDF through a nonlinear function will not be another Gaussian PDF. By keeping only the mean and covariance of the posterior PDF, we are approximating the posterior (by throwing away higher statistical moments).
- (ii) We are approximating the covariance of the true output PDF by linearizing the nonlinear function.
- (iii) The operating point about which we linearize the nonlinear function is often not the true mean of the prior PDF, but rather our estimate of the mean of the input PDF. This is an approximation that introduces error.
- (iv) We are approximating the mean of the true output PDF by simply passing the mean of the prior PDF through the nonlinear function. This does not represent the true mean of the output.

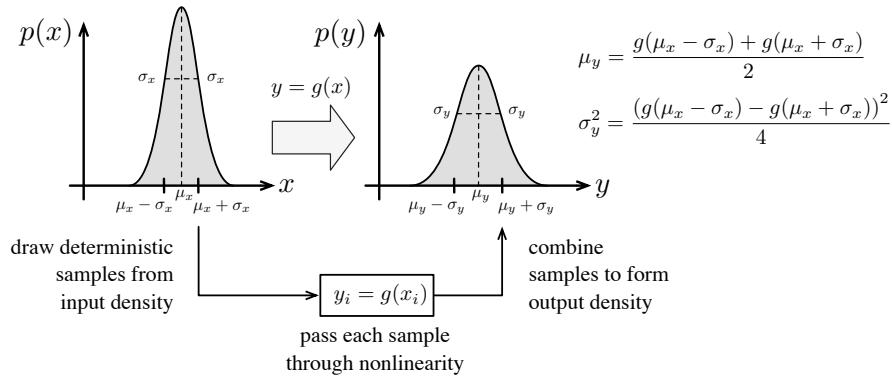
Another disadvantage of linearization is that we need to be able to either calculate the Jacobian of the nonlinearity in closed form, or compute it numerically (which introduces yet another approximation).

Despite all these approximations and disadvantages, if the function is only slightly nonlinear, and the input PDF is Gaussian, the linearization method is very simple to understand and quick to implement. One advantage⁸ is that the procedure is actually reversible (if the nonlinearity is locally invertible). That is, we can recover the input PDF exactly

⁸ It might be more accurate to say this is a by-product than an advantage, since it is a direct result of the specific approximations made in linearization.

Figure 4.9
One-dimensional Gaussian PDF transformed through a deterministic nonlinear function, $g(\cdot)$. Here we linearize the nonlinearity to propagate the variance approximately.

Figure 4.10
One-dimensional Gaussian PDF transformed through a deterministic nonlinear function, $g(\cdot)$. Here the basic sigmapoint transformation is used in which only two deterministic samples (one on either side of the mean) approximate the input density.



by passing the output PDF through the inverse of the nonlinearity (using the same linearization procedure). This is not true for all methods of passing PDFs through nonlinearities since they do not all make the same approximations as linearization. For example, the sigmapoint transformation is not reversible in this way.

Sigmapoint Transformation

In a sense, the *sigmapoint (SP)* or *unscented* transformation (Julier and Uhlmann, 1996) is the compromise between the Monte Carlo and linearization methods when the input density is roughly a Gaussian PDF. It is more accurate than linearization, but for a comparable computational cost to linearization. Monte Carlo is still the most accurate method, but the computational cost is prohibitive in most situations.

It is actually a bit misleading to refer to ‘the’ sigmapoint transformation, as there is actually a whole family of such transformations. Figure 4.10 depicts the very simplest version in one dimension. In general, a version of the SP transformation is used that includes one additional sample beyond the basic version at the mean of the input density. The steps are as follows:

1. A set of $2L + 1$ *sigmapoints* is computed from the input density, $\mathcal{N}(\boldsymbol{\mu}_x, \boldsymbol{\Sigma}_{xx})$, according to

$$\mathbf{L}\mathbf{L}^T = \boldsymbol{\Sigma}_{xx}, \quad (\text{Cholesky decomposition, } \mathbf{L} \text{ lower-triangular}) \quad (4.47a)$$

$$\mathbf{x}_0 = \boldsymbol{\mu}_x, \quad (4.47b)$$

$$\mathbf{x}_i = \boldsymbol{\mu}_x + \sqrt{L + \kappa} \text{col}_i \mathbf{L}, \quad i = 1 \dots L \quad (4.47c)$$

$$\mathbf{x}_{i+L} = \boldsymbol{\mu}_x - \sqrt{L + \kappa} \text{col}_i \mathbf{L}, \quad (4.47d)$$

where $L = \dim(\boldsymbol{\mu}_x)$. We note that

$$\boldsymbol{\mu}_x = \sum_{i=0}^{2L} \alpha_i \mathbf{x}_i, \quad (4.48a)$$

$$\boldsymbol{\Sigma}_{xx} = \sum_{i=0}^{2L} \alpha_i (\mathbf{x}_i - \boldsymbol{\mu}_x) (\mathbf{x}_i - \boldsymbol{\mu}_x)^T, \quad (4.48b)$$

where

$$\alpha_i = \begin{cases} \frac{\kappa}{L+\kappa} & i = 0 \\ \frac{1-\kappa}{2(L+\kappa)} & \text{otherwise} \end{cases}, \quad (4.49)$$

which we note sums to 1. The user-definable parameter, κ , will be explained in the next section.

2. Each of the sigmapoints is individually passed through the nonlinearity, $\mathbf{g}(\cdot)$:

$$\mathbf{y}_i = \mathbf{g}(\mathbf{x}_i), \quad i = 0 \dots 2L. \quad (4.50)$$

3. The mean of the output density, $\boldsymbol{\mu}_y$, is computed as

$$\boldsymbol{\mu}_y = \sum_{i=0}^{2L} \alpha_i \mathbf{y}_i. \quad (4.51)$$

4. The covariance of the output density, $\boldsymbol{\Sigma}_{yy}$, is computed as

$$\boldsymbol{\Sigma}_{yy} = \sum_{i=0}^{2L} \alpha_i (\mathbf{y}_i - \boldsymbol{\mu}_y) (\mathbf{y}_i - \boldsymbol{\mu}_y)^T. \quad (4.52)$$

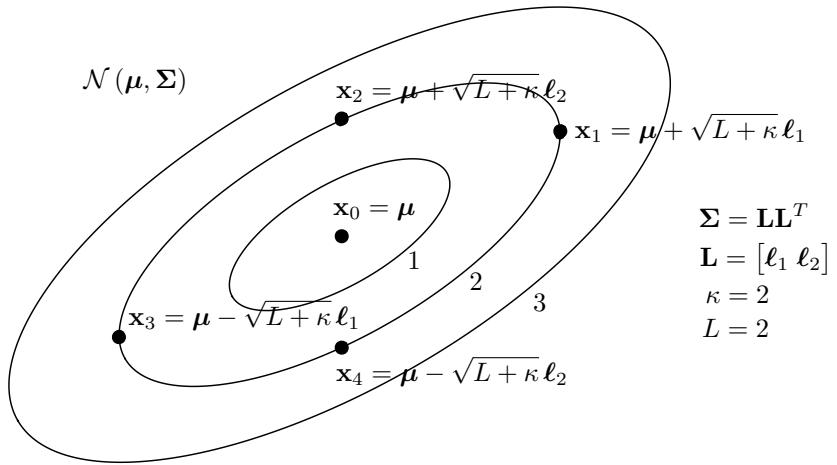
5. The output density, $\mathcal{N}(\boldsymbol{\mu}_y, \boldsymbol{\Sigma}_{yy})$, is returned.

This method of transforming a PDF through a nonlinearity has a number of advantages over linearization:

- (i) By approximating the input density instead of linearizing, we avoid the need to compute the Jacobian of the nonlinearity (either in closed form or numerically). Figure 4.11 provides an example of the sigmapoints for a two-dimensional Gaussian.
- (ii) We employ only standard linear algebra operations (Cholesky decomposition, outer products, matrix summations).
- (iii) The computation cost is similar to linearization (when a numerical Jacobian is used).
- (iv) There is no requirement that the nonlinearity be smooth and differentiable.

The next section will furthermore show that the unscented transformation can also more accurately capture the posterior density than linearization (by way of an example).

Figure 4.11
 Two-dimensional ($L = 2$) Gaussian PDF, whose covariance is displayed using elliptical equiprobable contours of 1, 2, and 3 standard deviations, and the corresponding $2L + 1 = 5$ sigmapoints for $\kappa = 2$.



Example 4.1 We will use a simple one-dimensional nonlinearity, $f(x) = x^2$, as an example and compare the various transformation methods. Let the prior density be $\mathcal{N}(\mu_x, \sigma_x^2)$.

Monte Carlo Method

In fact, for this particularly nonlinearity, we can essentially use the Monte Carlo method in closed form (i.e., we do not actually draw any samples) to get the exact answer for transforming the input density through the nonlinearity. An arbitrary sample (a.k.a., realization) of the input density is given by

$$x_i = \mu_x + \delta x_i, \quad \delta x_i \leftarrow \mathcal{N}(0, \sigma_x^2). \quad (4.53)$$

Transforming this sample through the nonlinearity, we get

$$y_i = f(x_i) = f(\mu_x + \delta x_i) = (\mu_x + \delta x_i)^2 = \mu_x^2 + 2\mu_x \delta x_i + \delta x_i^2. \quad (4.54)$$

Taking the expectation of both sides, we arrive at the mean of the output:

$$\mu_y = E[y_i] = \mu_x^2 + 2\mu_x \underbrace{E[\delta x_i]}_0 + \underbrace{E[\delta x_i^2]}_{\sigma_x^2} = \mu_x^2 + \sigma_x^2. \quad (4.55)$$

We do a similar thing for the variance of the output:

$$\sigma_y^2 = E[(y_i - \mu_y)^2] \quad (4.56a)$$

$$= E[(2\mu_x \delta x_i + \delta x_i^2 - \sigma_x^2)^2] \quad (4.56b)$$

$$= \underbrace{E[\delta x_i^4]}_{3\sigma_x^4} + 4\mu_x \underbrace{E[\delta x_i^3]}_0 + (4\mu_x^2 - 2\sigma_x^2) \underbrace{E[\delta x_i^2]}_{\sigma_x^2} \\ - 4\mu_x \sigma_x^2 \underbrace{E[\delta x_i]}_0 + \sigma_x^4 \quad (4.56c)$$

$$= 4\mu_x^2 \sigma_x^2 + 2\sigma_x^4, \quad (4.56d)$$

where $E[\delta x_i^3] = 0$ and $E[\delta x_i^4] = 3\sigma_x^4$ are the well-known third and fourth moments for a Gaussian PDF.

In truth, the resulting output density is not Gaussian. We could go on to compute higher moments of the output (and they would not all match a Gaussian). However, if we want to approximate the output as Gaussian by not considering the moments beyond the variance, we can. In this case, the resulting output density is $\mathcal{N}(\mu_y, \sigma_y^2)$. We have effectively used the Monte Carlo method with an infinite number of samples to carry out the computation of the first two moments of the posterior exactly in closed form. Let us now see how linearization and the sigmapoint transformation perform.

Linearization

Linearizing the nonlinearity about the mean of the input density, we have

$$y_i = f(\mu_x + \delta x_i) \approx \underbrace{f(\mu_x)}_{\mu_x^2} + \underbrace{\frac{\partial f}{\partial x} \Big|_{\mu_x}}_{2\mu_x} \delta x_i = \mu_x^2 + 2\mu_x \delta x_i. \quad (4.57)$$

Taking the expectation, we arrive at the mean of the output:

$$\mu_y = E[y_i] = \mu_x^2 + 2\mu_x \underbrace{E[\delta x_i]}_0 = \mu_x^2, \quad (4.58)$$

which is just the mean of the input passed through the nonlinearity: $\mu_y = f(\mu_x)$. For the variance of the output we have

$$\sigma_y^2 = E[(y_i - \mu_y)^2] = E[(2\mu_x \delta x_i)^2] = 4\mu_x^2 \sigma_x^2. \quad (4.59)$$

Comparing (4.55) with (4.58), and (4.56) with (4.59), we see there are some discrepancies. In fact, the linearized mean has a bias and the variance is too small (i.e., overconfident). Let us see what happens with the sigmapoint transformation.

Sigmapoint Transformation

There are $2L + 1 = 3$ sigmapoints in dimension $L = 1$:

$$x_0 = \mu_x, \quad x_1 = \mu_x + \sqrt{1 + \kappa} \sigma_x, \quad x_2 = \mu_x - \sqrt{1 + \kappa} \sigma_x, \quad (4.60)$$

where κ is a user-definable parameter that we discuss below. We pass each sigmapoint through the nonlinearity:

$$y_0 = f(x_0) = \mu_x^2, \quad (4.61a)$$

$$\begin{aligned} y_1 &= f(x_1) = (\mu_x + \sqrt{1 + \kappa} \sigma_x)^2 \\ &= \mu_x^2 + 2\mu_x \sqrt{1 + \kappa} \sigma_x + (1 + \kappa) \sigma_x^2, \end{aligned} \quad (4.61b)$$

$$\begin{aligned} y_2 &= f(x_2) = (\mu_x - \sqrt{1 + \kappa} \sigma_x)^2 \\ &= \mu_x^2 - 2\mu_x \sqrt{1 + \kappa} \sigma_x + (1 + \kappa) \sigma_x^2. \end{aligned} \quad (4.61c)$$

The mean of the output is given by

$$\mu_y = \frac{1}{1 + \kappa} \left(\kappa y_0 + \frac{1}{2} \sum_{i=1}^2 y_i \right) \quad (4.62a)$$

$$\begin{aligned} &= \frac{1}{1 + \kappa} \left(\kappa \mu_x^2 + \frac{1}{2} (\mu_x^2 + 2\mu_x \sqrt{1 + \kappa} \sigma_x + (1 + \kappa) \sigma_x^2 + \mu_x^2 \right. \\ &\quad \left. - 2\mu_x \sqrt{1 + \kappa} \sigma_x + (1 + \kappa) \sigma_x^2) \right) \end{aligned} \quad (4.62b)$$

$$= \frac{1}{1 + \kappa} (\kappa \mu_x^2 + \mu_x^2 + (1 + \kappa) \sigma_x^2) \quad (4.62c)$$

$$= \mu_x^2 + \sigma_x^2, \quad (4.62d)$$

which is independent of κ and exactly the same as (4.55). For the variance we have

$$\sigma_y^2 = \frac{1}{1 + \kappa} \left(\kappa (y_0 - \mu_y)^2 + \frac{1}{2} \sum_{i=1}^2 (y_i - \mu_y)^2 \right) \quad (4.63a)$$

$$\begin{aligned} &= \frac{1}{1 + \kappa} \left(\kappa \sigma_x^4 + \frac{1}{2} \left((2\mu_x \sqrt{1 + \kappa} \sigma_x + \kappa \sigma_x^2)^2 \right. \right. \\ &\quad \left. \left. + (-2\mu_x \sqrt{1 + \kappa} \sigma_x + \kappa \sigma_x^2)^2 \right) \right) \end{aligned} \quad (4.63b)$$

$$= \frac{1}{1 + \kappa} (\kappa \sigma_x^4 + 4(1 + \kappa) \mu_x^2 \sigma_x^2 + \kappa^2 \sigma_x^4) \quad (4.63c)$$

$$= 4\mu_x^2 \sigma_x^2 + \kappa \sigma_x^4, \quad (4.63d)$$

which can be made to be identical to (4.56) by selecting the user-definable parameter, κ , to be 2. Thus, for this nonlinearity, the unscented transformation can exactly capture the correct mean and variance of the output.

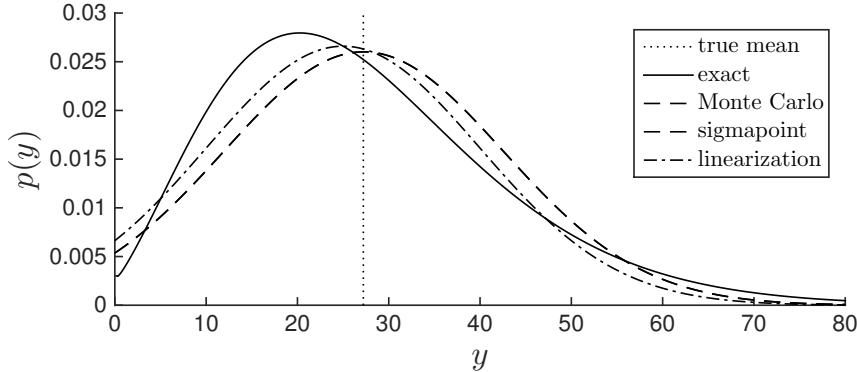


Figure 4.12
 Graphical depiction of passing a Gaussian PDF, $p(x) = \mathcal{N}(5, (3/2)^2)$, through the nonlinearity, $y = x^2$, using various methods. We see that the Monte Carlo and sigmapoint methods match the true mean, while linearization does not. We also show the exact transformed PDF, which is not Gaussian and therefore does not have its mean at its mode.

To understand why we should pick $\kappa = 2$, we need look no further than the input density. The parameter κ scales how far away the sigmapoints are from the mean. This does not affect the first three moments of the sigmapoints (i.e., μ_x , σ_x^2 , and the zero *skewness*). However, changing κ does influence the fourth moment, *kurtosis*. We already used the fact that for a Gaussian PDF, the fourth moment is $3\sigma_x^4$. We can choose κ to make the fourth moment of the sigmapoints match the true kurtosis of the Gaussian input density:

$$3\sigma_x^4 = \frac{1}{1+\kappa} \left(\underbrace{\kappa (x_0 - \mu_x)^4}_0 + \frac{1}{2} \sum_{i=1}^2 (x_i - \mu_x)^4 \right) \quad (4.64a)$$

$$= \frac{1}{2(1+\kappa)} \left((\sqrt{1+\kappa}\sigma_x)^4 + (-\sqrt{1+\kappa}\sigma_x)^4 \right) \quad (4.64b)$$

$$= (1+\kappa)\sigma_x^4. \quad (4.64c)$$

Comparing the desired and actual kurtosis, we should pick $\kappa = 2$ to make them match exactly. Not surprisingly, this has a positive effect on accuracy of the transformation.

In summary, this example shows that linearization is an inferior method of transforming a PDF through a nonlinearity if the goal is to capture the true mean of the output. Figure 4.12 provides a graphical depiction of this example.

In the next few sections, we return to the Bayes filter and use our new knowledge about the different methods of passing PDFs through nonlinearities to make some useful improvements to the EKF. We will begin with the particle filter, which makes use of the Monte Carlo method. We will then try to implement a Gaussian filter using the SP transformation.

4.2.8 Particle Filter

We have seen above that drawing a large number of samples is one way to approximate a PDF. We further saw that we could pass each sample through a nonlinearity and recombine them on the other side to get an approximation of the transformation of a PDF. In this section, we extend this idea to an approximation of the Bayes filter, called the *particle filter* (Thrun et al., 2001).

The particle filter is one of the only practical techniques able to handle non-Gaussian noise and nonlinear observation and motion models. It is practical in that it is very easy to implement; we do not even need to have analytical expressions for $\mathbf{f}(\cdot)$ and $\mathbf{g}(\cdot)$, nor for their derivatives.

There are actually many different flavours of the particle filter; we will outline a basic version and indicate where the variations typically occur. The approach taken here is based on *sample importance resampling* where the so-called *proposal* PDF is the prior PDF in the Bayes filter, propagated forward using the motion model and the latest motion measurement, \mathbf{v}_k . This version of the particle filter is sometimes called the *bootstrap algorithm*, the *condensation algorithm*, or the *survival-of-the-fittest algorithm*.

PF Algorithm

Using the notation from the section on the Bayes filter, the main steps in the particle filter are as follows:

1. Draw M samples from the joint density comprising the prior and the motion noise:

$$\begin{bmatrix} \hat{\mathbf{x}}_{k-1,m} \\ \mathbf{w}_{k,m} \end{bmatrix} \leftarrow p(\mathbf{x}_{k-1} | \check{\mathbf{x}}_0, \mathbf{v}_{1:k-1}, \mathbf{y}_{1:k-1}) p(\mathbf{w}_k), \quad (4.65)$$

where m is the unique particle index. In practice we can just draw from each factor of this joint density separately.

2. Generate a prediction of the posterior PDF by using \mathbf{v}_k . This is done by passing each prior particle/noise sample through the nonlinear motion model:

$$\check{\mathbf{x}}_{k,m} = \mathbf{f}(\hat{\mathbf{x}}_{k-1,m}, \mathbf{v}_k, \mathbf{w}_{k,m}). \quad (4.66)$$

These new ‘predicted particles’ together approximate the density, $p(\mathbf{x}_k | \check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{1:k-1})$.

3. Correct the posterior PDF by incorporating \mathbf{y}_k . This is done indirectly in two steps:

- First, assign a scalar weight, $w_{k,m}$, to each predicted particle based on the divergence between the desired posterior and the predicted posterior for each particle:

$$w_{k,m} = \frac{p(\check{\mathbf{x}}_{k,m} | \check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{1:k})}{p(\check{\mathbf{x}}_{k,m} | \check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{1:k-1})} = \eta p(\mathbf{y}_k | \check{\mathbf{x}}_{k,m}), \quad (4.67)$$

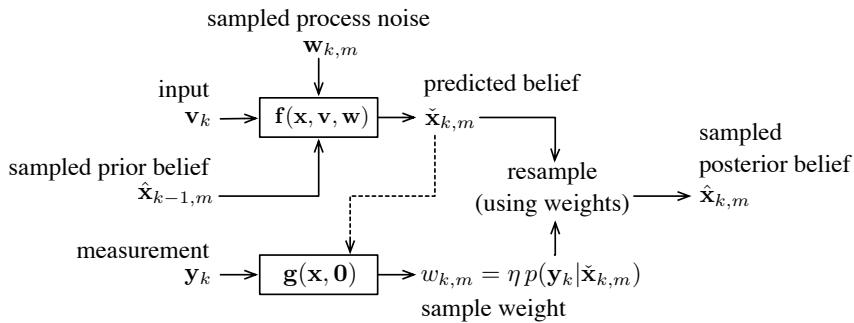


Figure 4.13
Block-diagram representation of particle filter.

where η is a normalization constant. This is typically accomplished in practice by simulating an expected sensor reading, $\check{y}_{k,m}$, using the nonlinear observation model:

$$\check{y}_{k,m} = \mathbf{g}(\check{x}_{k,m}, \mathbf{0}). \quad (4.68)$$

We then assume $p(\mathbf{y}_k | \check{x}_{k,m}) = p(\mathbf{y}_k | \check{y}_{k,m})$, where the right-hand side is a known density (e.g., Gaussian).

- Resample the posterior based on the weight assigned to each predicted posterior particle:

$$\hat{x}_{k,m} \xleftarrow{\text{resample}} \{\check{x}_{k,m}, w_{k,m}\}. \quad (4.69)$$

This can be done in several different ways. Madow provides a simple systematic technique to do resampling, which we describe below.

Figure 4.13 captures these steps in a block diagram.

Some additional comments should be made at this point to help get this basic version of the particle filter working in practical situations:

- (i) How do we know how many particles to use? It depends very much on the specific estimation problem. Typically hundreds of particles are used for low-dimensional problems (e.g., $\mathbf{x} = (x, y, \theta)$).
- (ii) We can dynamically pick the number of particles online using a heuristic such as $\sum w_{k,m} \geq w_{\text{thresh}}$, a threshold. In other words, we keep adding particles/samples and repeating steps 1 through 3 until the sum of the weights exceeds an experimentally determined threshold.
- (iii) We do not necessarily need to resample every time we go through the algorithm. We can delay resampling, but then need to carry the weights forward to the next iteration of the algorithm.
- (iv) To be on the safe side, it is wise to add a small percentage of samples in during Step 1 that are uniformly drawn from the entire state sample space. This protects against outlier sensor measurements/vehicle movements.

- (v) For high-dimensional state estimation problems, the particle filter can become computationally intractable. If too few particles are used, the densities involved are undersampled and give highly skewed results. The number of samples needed goes up exponentially with the dimension of the state space. Thrun et al. (2001) offer some alternative flavours of the particle filter to combat sample impoverishment.
- (vi) The particle filter is an ‘anytime’ algorithm. That is, we can just keep adding particles until we run out of time, then resample and give an answer. Using more particles always helps, but comes with a computational cost.
- (vii) The *Cramér-Rao lower bound (CRLB)* is set by the uncertainty in the measurements that we have available. Using more samples does not allow us to do better than the CRLB. See Section 2.2.11 for some discussion of the CRLB.

Resampling

A key aspect of particle filters is the need to resample the posterior density according to weights assigned to each current sample. One way to do this is to use the systematic resampling method described by Madow (1949). We assume we have M samples and that each of these is assigned an unnormalized weight, $w_m \in \mathbb{R} > 0$. From the weights, we create bins with boundaries, β_m , according to

$$\beta_m = \frac{\sum_{n=1}^m w_n}{\sum_{\ell=1}^M w_\ell}. \quad (4.70)$$

The β_m define the boundaries of M bins on the interval $[0, 1]$:

$$0 \leq \beta_1 \leq \beta_2 \leq \dots \leq \beta_{M-1} \leq 1,$$

where we note that we will have $\beta_M \equiv 1$. We then select a random number, ρ , sampled from a uniform density on $[0, 1)$. For M iterations we add to the new list of samples, the sample whose bin contains ρ . At each iteration we step ρ forward by $1/M$. The algorithm guarantees that all bins whose size is greater than $1/M$ will have a sample in the new list.

4.2.9 Sigmapoint Kalman Filter

Another way we can attempt to improve on the basic EKF is to get rid of the idea of linearizing altogether and instead use the sigmapoint transformation to pass PDFs through the nonlinear motion and observation models. The result is the *sigmapoint Kalman filter (SPKF)*, also

sometimes called the *unscented Kalman filter (UKF)*. We will discuss the prediction and correction steps separately⁹:

Prediction Step

The prediction step is a fairly straightforward application of the sigma-point transformation since we are simply trying to bring our prior forward in time through the motion model. We employ the following steps to go from the prior belief, $\{\hat{\mathbf{x}}_{k-1}, \hat{\mathbf{P}}_{k-1}\}$, to the predicted belief, $\{\check{\mathbf{x}}_k, \check{\mathbf{P}}_k\}$:

1. Both the prior belief and the motion noise have uncertainty, so these are stacked together in the following way:

$$\boldsymbol{\mu}_z = \begin{bmatrix} \hat{\mathbf{x}}_{k-1} \\ \mathbf{0} \end{bmatrix}, \quad \boldsymbol{\Sigma}_{zz} = \begin{bmatrix} \hat{\mathbf{P}}_{k-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_k \end{bmatrix}, \quad (4.71)$$

where we see that $\{\boldsymbol{\mu}_z, \boldsymbol{\Sigma}_{zz}\}$ is still a Gaussian representation. We let $L = \dim \boldsymbol{\mu}_z$.

2. Convert $\{\boldsymbol{\mu}_z, \boldsymbol{\Sigma}_{zz}\}$ to a sigmapoint representation:

$$\mathbf{L}\mathbf{L}^T = \boldsymbol{\Sigma}_{zz}, \quad (\text{Cholesky decomposition, } \mathbf{L} \text{ lower-triangular}) \quad (4.72a)$$

$$\mathbf{z}_0 = \boldsymbol{\mu}_z, \quad (4.72b)$$

$$\mathbf{z}_i = \boldsymbol{\mu}_z + \sqrt{L + \kappa} \text{col}_i \mathbf{L}, \quad i = 1 \dots L \quad (4.72c)$$

$$\mathbf{z}_{i+L} = \boldsymbol{\mu}_z - \sqrt{L + \kappa} \text{col}_i \mathbf{L}. \quad (4.72d)$$

3. Unstack each sigmapoint into state and motion noise,

$$\mathbf{z}_i = \begin{bmatrix} \hat{\mathbf{x}}_{k-1,i} \\ \mathbf{w}_{k,i} \end{bmatrix}, \quad (4.73)$$

and then pass each sigmapoint through the nonlinear motion model exactly:

$$\check{\mathbf{x}}_{k,i} = \mathbf{f}(\hat{\mathbf{x}}_{k-1,i}, \mathbf{v}_k, \mathbf{w}_{k,i}), \quad i = 0 \dots 2L. \quad (4.74)$$

Note that the latest input, \mathbf{v}_k , is required.

4. Recombine the transformed sigmapoints into the predicted belief, $\{\check{\mathbf{x}}_k, \check{\mathbf{P}}_k\}$, according to

$$\check{\mathbf{x}}_k = \sum_{i=0}^{2L} \alpha_i \check{\mathbf{x}}_{k,i}, \quad (4.75a)$$

$$\check{\mathbf{P}}_k = \sum_{i=0}^{2L} \alpha_i (\check{\mathbf{x}}_{k,i} - \check{\mathbf{x}}_k) (\check{\mathbf{x}}_{k,i} - \check{\mathbf{x}}_k)^T, \quad (4.75b)$$

⁹ These are sometimes handled together in a single step, but we prefer to think of each of these as a separate application of the sigmapoint transformation.

where

$$\alpha_i = \begin{cases} \frac{\kappa}{L+\kappa} & i=0 \\ \frac{1}{2} \frac{1}{L+\kappa} & \text{otherwise} \end{cases} . \quad (4.76)$$

Next we will look at a second application of the sigmapoint transformation to implement the correction step.

Correction Step

This step is a little more complicated. We look back to Section 4.2.4 and recall that the conditional Gaussian density for \mathbf{x}_k (i.e., the posterior) is

$$\begin{aligned} p(\mathbf{x}_k | \check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k}) \\ = \mathcal{N}\left(\underbrace{\boldsymbol{\mu}_{x,k} + \boldsymbol{\Sigma}_{xy,k} \boldsymbol{\Sigma}_{yy,k}^{-1} (\mathbf{y}_k - \boldsymbol{\mu}_{y,k})}_{\hat{\mathbf{x}}_k}, \underbrace{\boldsymbol{\Sigma}_{xx,k} - \boldsymbol{\Sigma}_{xy,k} \boldsymbol{\Sigma}_{yy,k}^{-1} \boldsymbol{\Sigma}_{yx,k}}_{\hat{\mathbf{P}}_k}\right), \end{aligned} \quad (4.77)$$

where we have defined $\hat{\mathbf{x}}_k$ as the mean and $\hat{\mathbf{P}}_k$ as the covariance. In this form, we can write the generalized Gaussian correction-step equations as

$$\mathbf{K}_k = \boldsymbol{\Sigma}_{xy,k} \boldsymbol{\Sigma}_{yy,k}^{-1}, \quad (4.78a)$$

$$\hat{\mathbf{P}}_k = \check{\mathbf{P}}_k - \mathbf{K}_k \boldsymbol{\Sigma}_{xy,k}^T, \quad (4.78b)$$

$$\hat{\mathbf{x}}_k = \check{\mathbf{x}}_k + \mathbf{K}_k (\mathbf{y}_k - \boldsymbol{\mu}_{y,k}). \quad (4.78c)$$

We will use the SP transformation to come up with better versions of $\boldsymbol{\mu}_{y,k}$, $\boldsymbol{\Sigma}_{yy,k}$, and $\boldsymbol{\Sigma}_{xy,k}$. We employ the following steps:

1. Both the predicted belief and the observation noise have uncertainty, so these are stacked together in the following way:

$$\boldsymbol{\mu}_z = \begin{bmatrix} \check{\mathbf{x}}_k \\ \mathbf{0} \end{bmatrix}, \quad \boldsymbol{\Sigma}_{zz} = \begin{bmatrix} \check{\mathbf{P}}_k & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_k \end{bmatrix}, \quad (4.79)$$

where we see that $\{\boldsymbol{\mu}_z, \boldsymbol{\Sigma}_{zz}\}$ is still a Gaussian representation. We let $L = \dim \boldsymbol{\mu}_z$.

2. Convert $\{\boldsymbol{\mu}_z, \boldsymbol{\Sigma}_{zz}\}$ to a sigmapoint representation:

$$\mathbf{LL}^T = \boldsymbol{\Sigma}_{zz}, \quad (\text{Cholesky decomposition, } \mathbf{L} \text{ lower-triangular}) \quad (4.80a)$$

$$\mathbf{z}_0 = \boldsymbol{\mu}_z, \quad (4.80b)$$

$$\mathbf{z}_i = \boldsymbol{\mu}_z + \sqrt{L+\kappa} \text{ col}_i \mathbf{L}, \quad (4.80c)$$

$$\mathbf{z}_{i+L} = \boldsymbol{\mu}_z - \sqrt{L+\kappa} \text{ col}_i \mathbf{L}. \quad (4.80d)$$

3. Unstack each sigmapoint into state and observation noise,

$$\mathbf{z}_i = \begin{bmatrix} \check{\mathbf{x}}_{k,i} \\ \mathbf{n}_{k,i} \end{bmatrix}, \quad (4.81)$$

and then pass each sigmapoint through the nonlinear observation model exactly:

$$\check{\mathbf{y}}_{k,i} = \mathbf{g}(\check{\mathbf{x}}_{k,i}, \mathbf{n}_{k,i}). \quad (4.82)$$

4. Recombine the transformed sigmapoints into the desired moments:

$$\boldsymbol{\mu}_{y,k} = \sum_{i=0}^{2L} \alpha_i \check{\mathbf{y}}_{k,i}, \quad (4.83a)$$

$$\boldsymbol{\Sigma}_{yy,k} = \sum_{i=0}^{2L} \alpha_i (\check{\mathbf{y}}_{k,i} - \boldsymbol{\mu}_{y,k}) (\check{\mathbf{y}}_{k,i} - \boldsymbol{\mu}_{y,k})^T, \quad (4.83b)$$

$$\boldsymbol{\Sigma}_{xy,k} = \sum_{i=0}^{2L} \alpha_i (\check{\mathbf{x}}_{k,i} - \check{\mathbf{x}}_k) (\check{\mathbf{y}}_{k,i} - \boldsymbol{\mu}_{y,k})^T, \quad (4.83c)$$

where

$$\alpha_i = \begin{cases} \frac{\kappa}{L+\kappa} & i = 0 \\ \frac{1}{2} \frac{1}{L+\kappa} & \text{otherwise} \end{cases}. \quad (4.84)$$

These are plugged into the generalized Gaussian correction-step equations above to complete the correction step.

Two advantages of the SPKF are that it (i) does not require any analytical derivatives and (ii) uses only basic linear algebra operations in the implementation. Moreover, we do not even need the nonlinear motion and observation models in closed form; they could just be black-box software functions.

Comparing Terms to EKF

We see in the correction step that the matrix $\boldsymbol{\Sigma}_{yy,k}$ takes on the role of $\mathbf{G}_k \check{\mathbf{P}}_k \mathbf{G}_k^T + \mathbf{R}'_k$ in the EKF. We can see this more directly by linearizing the observation model (about the predicted state) as in the EKF:

$$\check{\mathbf{y}}_{k,i} = \mathbf{g}(\check{\mathbf{x}}_{k,i}, \mathbf{n}_{k,i}) \approx \mathbf{g}(\check{\mathbf{x}}_k, \mathbf{0}) + \mathbf{G}_k (\check{\mathbf{x}}_{k,i} - \check{\mathbf{x}}_k) + \mathbf{n}'_{k,i}. \quad (4.85)$$

Substituting this approximation into (4.83a), we can see that

$$\check{\mathbf{y}}_{k,i} - \boldsymbol{\mu}_{y,k} \approx \mathbf{G}_k (\check{\mathbf{x}}_{k,i} - \check{\mathbf{x}}_k) + \mathbf{n}'_{k,i}. \quad (4.86)$$

Substituting this into (4.83b), we have that

$$\begin{aligned} \boldsymbol{\Sigma}_{yy,k} &\approx \mathbf{G}_k \underbrace{\sum_{i=0}^{2L} \alpha_i (\check{\mathbf{x}}_{k,i} - \check{\mathbf{x}}_k) (\check{\mathbf{x}}_{k,i} - \check{\mathbf{x}}_k)^T}_{\check{\mathbf{P}}_k} \mathbf{G}_k^T + \underbrace{\sum_{i=0}^{2L} \alpha_i \mathbf{n}'_{k,i} \mathbf{n}'_{k,i}^T}_{\mathbf{R}'_k} \\ &+ \mathbf{G}_k \underbrace{\sum_{i=0}^{2L} \alpha_i (\check{\mathbf{x}}_{k,i} - \check{\mathbf{x}}_k) \mathbf{n}'_{k,i}^T}_{0} + \underbrace{\sum_{i=0}^{2L} \alpha_i \mathbf{n}'_{k,i} (\check{\mathbf{x}}_{k,i} - \check{\mathbf{x}}_k)^T}_{0} \mathbf{G}_k^T, \quad (4.87) \end{aligned}$$

where some of the terms are zero owing to the block-diagonal structure of Σ_{zz} above. For $\Sigma_{xy,k}$, by substituting our approximation into (4.83c), we have

$$\Sigma_{xy,k} \approx \underbrace{\sum_{i=0}^{2L} \alpha_i (\check{\mathbf{x}}_{k,i} - \check{\mathbf{x}}_k) (\check{\mathbf{x}}_{k,i} - \check{\mathbf{x}}_k)^T \mathbf{G}_k^T}_{\check{\mathbf{P}}_k} + \underbrace{\sum_{i=0}^{2L} \alpha_i (\check{\mathbf{x}}_{k,i} - \check{\mathbf{x}}_k) \mathbf{n}'_{k,i}^T}_{0}, \quad (4.88)$$

so that

$$\mathbf{K}_k = \Sigma_{xy,k} \Sigma_{yy,k}^{-1} \approx \check{\mathbf{P}}_k \mathbf{G}_k^T (\mathbf{G}_k \check{\mathbf{P}}_k \mathbf{G}_k^T + \mathbf{R}'_k)^{-1}, \quad (4.89)$$

which is what we had in the EKF.

Special Case of Linear Dependence on Measurement Noise

In the case that our nonlinear observation model has the special form

$$\mathbf{y}_k = \mathbf{g}(\mathbf{x}_k) + \mathbf{n}_k, \quad (4.90)$$

the SPKF correction step can be greatly sped up. Without loss of generality, we can break the sigmapoints into two categories based on the block-diagonal partitioning in the matrix, Σ_{zz} , above; we say there are $2N + 1$ sigmapoints coming from the dimension of the state and $2(L - N)$ additional sigmapoints coming from the dimension of the measurements. To make this convenient, we will re-order the indexing on the sigmapoints accordingly:

$$\check{\mathbf{y}}_{k,j} = \begin{cases} \mathbf{g}(\check{\mathbf{x}}_{k,j}) & j = 0 \dots 2N \\ \mathbf{g}(\check{\mathbf{x}}_k) + \mathbf{n}_{k,j} & j = 2N + 1 \dots 2L + 1 \end{cases}. \quad (4.91)$$

We can then write our expression for $\mu_{y,k}$ as

$$\mu_{y,k} = \sum_{j=0}^{2N} \alpha_j \check{\mathbf{y}}_{k,j} + \sum_{j=2N+1}^{2L+1} \alpha_j \check{\mathbf{y}}_{k,j} \quad (4.92a)$$

$$= \sum_{j=0}^{2N} \alpha_j \check{\mathbf{y}}_{k,j} + \sum_{j=2N+1}^{2L+1} \alpha_j (\mathbf{g}(\check{\mathbf{x}}_k) + \mathbf{n}_{k,j}) \quad (4.92b)$$

$$= \sum_{j=0}^{2N} \alpha_j \check{\mathbf{y}}_{k,j} + \mathbf{g}(\check{\mathbf{x}}_k) \sum_{j=2N+1}^{2L+1} \alpha_j \quad (4.92c)$$

$$= \sum_{j=0}^{2N} \beta_j \check{\mathbf{y}}_{k,j}, \quad (4.92d)$$

where

$$\beta_i = \begin{cases} \alpha_i + \sum_{j=2N+1}^{2L+1} \alpha_j & i = 0 \\ \alpha_i & \text{otherwise} \end{cases} \quad (4.93a)$$

$$= \begin{cases} \frac{(\kappa+L-N)}{N+(\kappa+L-N)} & i = 0 \\ \frac{1}{2} \frac{1}{N+(\kappa+L-N)} & \text{otherwise} \end{cases}. \quad (4.93b)$$

The is the same form as the original weights (and they still sum to 1). We can then easily verify that

$$\Sigma_{yy,k} = \sum_{j=0}^{2N} \beta_j (\check{\mathbf{y}}_{k,j} - \boldsymbol{\mu}_{y,k}) (\check{\mathbf{y}}_{k,j} - \boldsymbol{\mu}_{y,k})^T + \mathbf{R}_k, \quad (4.94)$$

with no approximation. This is already helpful in that we do not really need all $2L + 1$ sigmapoints but only $2N + 1$ of them. This means we do not need to call $\mathbf{g}(\cdot)$ as many times, which can be expensive in some situations.

We still have a problem, however. It is still necessary to invert $\Sigma_{yy,k}$, which is size $(L - N) \times (L - N)$, to compute the Kalman gain matrix. If the number of measurements, $L - N$, is large, this could be very expensive. We can make further gains if we assume that the inverse of \mathbf{R}_k can be computed cheaply. For example, if $\mathbf{R}_k = \sigma^2 \mathbf{1}$ then $\mathbf{R}_k^{-1} = \sigma^{-2} \mathbf{1}$. We proceed by noting that $\Sigma_{yy,k}$ can be conveniently written as

$$\Sigma_{yy,k} = \mathbf{Z}_k \mathbf{Z}_k^T + \mathbf{R}_k, \quad (4.95)$$

where

$$\text{col}_j \mathbf{Z}_k = \sqrt{\beta_j} (\check{\mathbf{y}}_{k,j} - \boldsymbol{\mu}_{y,k}). \quad (4.96)$$

By the SMW identity from Section 2.2.7, we can then show that

$$\Sigma_{yy,k}^{-1} = (\mathbf{Z}_k \mathbf{Z}_k^T + \mathbf{R}_k)^{-1} \quad (4.97a)$$

$$= \mathbf{R}_k^{-1} - \mathbf{R}_k^{-1} \mathbf{Z}_k \underbrace{(\mathbf{Z}_k^T \mathbf{R}_k^{-1} \mathbf{Z}_k + \mathbf{1})^{-1}}_{(2N+1) \times (2N+1)} \mathbf{Z}_k^T \mathbf{R}_k^{-1}, \quad (4.97b)$$

where we now only need to invert a $(2N+1) \times (2N+1)$ matrix (assuming \mathbf{R}_k^{-1} is known).

4.2.10 Iterated Sigmapoint Kalman Filter

An *iterated sigmapoint Kalman filter (ISPKF)* has been proposed by Sibley et al. (2006) that does better than the one-shot version. In this case, we compute input sigmapoints around an operating point, $\mathbf{x}_{\text{op},k}$, at each iteration. At the first iteration, we let $\mathbf{x}_{\text{op},k} = \check{\mathbf{x}}_k$, but this is then improved with each subsequent iteration. We show all the steps to avoid confusion:

- Both the predicted belief and the observation noise have uncertainty, so these are stacked together in the following way:

$$\boldsymbol{\mu}_z = \begin{bmatrix} \mathbf{x}_{\text{op},k} \\ \mathbf{0} \end{bmatrix}, \quad \boldsymbol{\Sigma}_{zz} = \begin{bmatrix} \check{\mathbf{P}}_k & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_k \end{bmatrix}, \quad (4.98)$$

where we see that $\{\boldsymbol{\mu}_z, \boldsymbol{\Sigma}_{zz}\}$ is still a Gaussian representation. We let $L = \dim \boldsymbol{\mu}_z$.

- Convert $\{\boldsymbol{\mu}_z, \boldsymbol{\Sigma}_{zz}\}$ to a sigmapoint representation:

$$\mathbf{L}\mathbf{L}^T = \boldsymbol{\Sigma}_{zz}, \quad (\text{Cholesky decomposition, } \mathbf{L} \text{ lower-triangular}) \quad (4.99a)$$

$$\mathbf{z}_0 = \boldsymbol{\mu}_z, \quad (4.99b)$$

$$\mathbf{z}_i = \boldsymbol{\mu}_z + \sqrt{L + \kappa} \text{col}_i \mathbf{L}, \quad i = 1 \dots L \quad (4.99c)$$

$$\mathbf{z}_{i+L} = \boldsymbol{\mu}_z - \sqrt{L + \kappa} \text{col}_i \mathbf{L}. \quad (4.99d)$$

- Unstack each sigmapoint into state and observation noise,

$$\mathbf{z}_i = \begin{bmatrix} \mathbf{x}_{\text{op},k,i} \\ \mathbf{n}_{k,i} \end{bmatrix}, \quad (4.100)$$

and then pass each sigmapoint through the nonlinear observation model exactly:

$$\mathbf{y}_{\text{op},k,i} = \mathbf{g}(\mathbf{x}_{\text{op},k,i}, \mathbf{n}_{k,i}). \quad (4.101)$$

- Recombine the transformed sigmapoints into the desired moments:

$$\boldsymbol{\mu}_{y,k} = \sum_{i=0}^{2L} \alpha_i \mathbf{y}_{\text{op},k,i}, \quad (4.102a)$$

$$\boldsymbol{\Sigma}_{yy,k} = \sum_{i=0}^{2L} \alpha_i (\mathbf{y}_{\text{op},k,i} - \boldsymbol{\mu}_{y,k}) (\mathbf{y}_{\text{op},k,i} - \boldsymbol{\mu}_{y,k})^T, \quad (4.102b)$$

$$\boldsymbol{\Sigma}_{xy,k} = \sum_{i=0}^{2L} \alpha_i (\mathbf{x}_{\text{op},k,i} - \mathbf{x}_{\text{op},k}) (\mathbf{y}_{\text{op},k,i} - \boldsymbol{\mu}_{y,k})^T. \quad (4.102c)$$

$$\boldsymbol{\Sigma}_{xx,k} = \sum_{i=0}^{2L} \alpha_i (\mathbf{x}_{\text{op},k,i} - \mathbf{x}_{\text{op},k}) (\mathbf{x}_{\text{op},k,i} - \mathbf{x}_{\text{op},k})^T, \quad (4.102d)$$

where

$$\alpha_i = \begin{cases} \frac{\kappa}{L+\kappa} & i = 0 \\ \frac{1}{2} \frac{1}{L+\kappa} & \text{otherwise} \end{cases}. \quad (4.103)$$

At this point, Sibley et al. use the relationships between the SPKF and EKF quantities to update the IEKF correction equations, (4.45), using

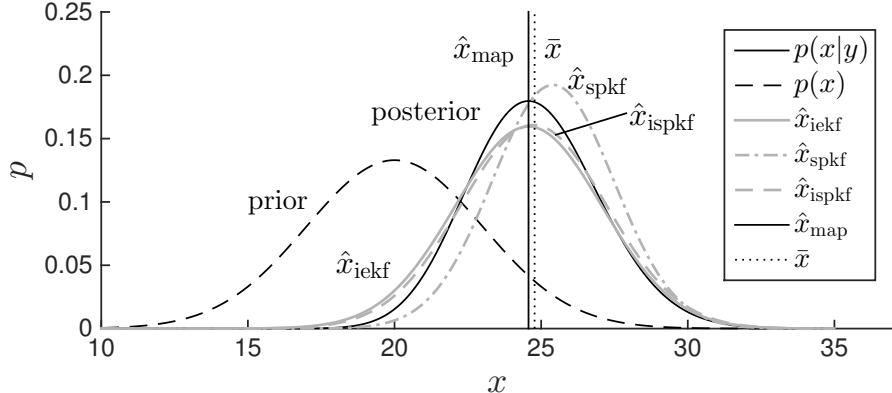


Figure 4.14
Stereo camera example, comparing the inference (i.e., ‘corrective’) step of the IEKF, SPKF, and ISPKF to the full Bayesian posterior, $p(x|y)$. We see that neither of the sigma-point methods matches up against the MAP solution, \hat{x}_{map} . Superficially, the ISPKF seems to come closer to the mean of the full posterior, \bar{x} .

the statistical rather than analytical Jacobian quantities:

$$\mathbf{K}_k = \underbrace{\check{\mathbf{P}}_k \mathbf{G}_k^T}_{\Sigma_{xy,k}} \left(\underbrace{\mathbf{G}_k \check{\mathbf{P}}_k \mathbf{G}_k^T}_{\Sigma_{yy,k}} + \mathbf{R}'_k \right)^{-1}, \quad (4.104a)$$

$$\hat{\mathbf{P}}_k = \left(\mathbf{1} - \mathbf{K}_k \underbrace{\mathbf{G}_k}_{\Sigma_{yx,k} \Sigma_{xx,k}^{-1}} \right) \underbrace{\check{\mathbf{P}}_k}_{\Sigma_{xx,k}}, \quad (4.104b)$$

$$\hat{\mathbf{x}}_k = \check{\mathbf{x}}_k + \mathbf{K}_k \left(\mathbf{y}_k - \underbrace{\mathbf{g}(\mathbf{x}_{\text{op},k}, \mathbf{0})}_{\mu_{y,k}} - \underbrace{\mathbf{G}_k}_{\Sigma_{yx,k} \Sigma_{xx,k}^{-1}} (\check{\mathbf{x}}_k - \mathbf{x}_{\text{op},k}) \right), \quad (4.104c)$$

which results in

$$\mathbf{K}_k = \Sigma_{xy,k} \Sigma_{yy,k}^{-1}, \quad (4.105a)$$

$$\hat{\mathbf{P}}_k = \Sigma_{xx,k} - \mathbf{K}_k \Sigma_{yx,k}, \quad (4.105b)$$

$$\hat{\mathbf{x}}_k = \check{\mathbf{x}}_k + \mathbf{K}_k \left(\mathbf{y}_k - \mu_{y,k} - \Sigma_{yx,k} \Sigma_{xx,k}^{-1} (\check{\mathbf{x}}_k - \mathbf{x}_{\text{op},k}) \right). \quad (4.105c)$$

Initially, we set the operating point to be the mean of the prior: $\mathbf{x}_{\text{op},k} = \check{\mathbf{x}}_k$. At subsequent iterations we set it to be the best estimate so far:

$$\mathbf{x}_{\text{op},k} \leftarrow \hat{\mathbf{x}}_k. \quad (4.106)$$

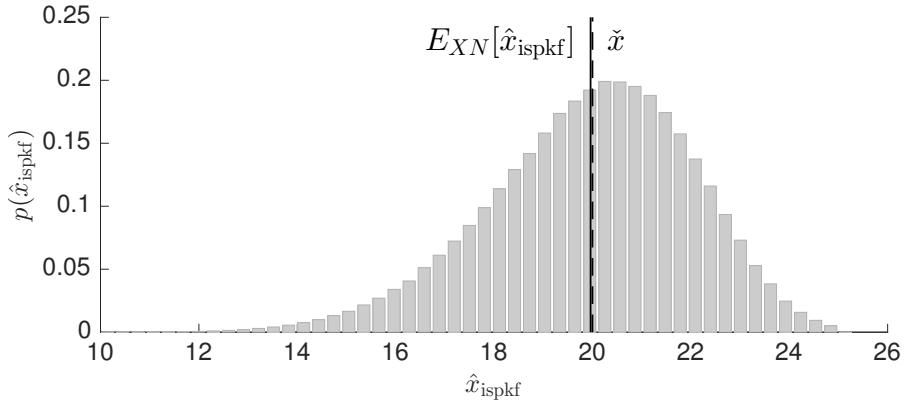
The process terminates when the change from one iteration to the next becomes sufficiently small.

We have seen that the first iteration of the IEKF results in the EKF method, and this is also true for the SPKF/ ISPKF. Setting $\mathbf{x}_{\text{op},k} = \check{\mathbf{x}}_k$ in (4.105c) results in

$$\hat{\mathbf{x}}_k = \check{\mathbf{x}}_k + \mathbf{K}_k (\mathbf{y}_k - \mu_{y,k}), \quad (4.107)$$

which is the same as the one-shot method in (4.78c).

Figure 4.15
 Histogram of estimator values for 1,000,000 trials of the stereo camera experiment where each time a new x_{true} is randomly drawn from the prior and a new y_{meas} is randomly drawn from the measurement model. The dashed line marks the mean of the prior, \bar{x} , and the solid line marks the expected value of the iterated sigma-point estimator, \hat{x}_{ispkf} , over all the trials. The gap between dashed and solid is $e_{\text{mean}} \approx -3.84$ cm, which indicates a small bias, and the average squared error is $e_{\text{sq}} \approx 4.32$ m².



4.2.11 ISPKF Seeks the Posterior Mean

Now, the question we must ask ourselves is, *how do the sigma-point estimates relate to the full posterior?* Figure 4.14 compares the sigma-point methods to the full posterior and iterated linearization (i.e., MAP) on our stereo camera example where we used

$$x_{\text{true}} = 26 \text{ [m]}, \quad y_{\text{meas}} = \frac{fb}{x_{\text{true}}} - 0.6 \text{ [pixel]},$$

again to exaggerate the differences between the various methods. Our implementations of the sigma-point methods used $\kappa = 2$, which is appropriate for a Gaussian prior. Much like the EKF, we see that the one-shot SPKF method bears no obvious relationship to the full posterior. However, the ISPKF method appears to come closer to the *mean*, \bar{x} , rather than the mode (i.e., MAP) value, \hat{x}_{map} . Numerically, the numbers of interest are:

$$\begin{aligned} \hat{x}_{\text{map}} &= 24.5694, & \bar{x} &= 24.7770, \\ \hat{x}_{\text{iekf}} &= 24.5694, & \hat{x}_{\text{ispkf}} &= 24.7414. \end{aligned}$$

We see that the IEKF solution matches the MAP one and that the ISPKF solution is close to (but not exactly) the mean.

Now, we consider the question, *how well does the iterated sigma-point method capture x_{true} ?* Once again, we compute the performance over a large number of trials (using the parameters in (4.5)). The results are shown in Figure 4.15. We see that the average difference of the estimator, \hat{x}_{ispkf} , and the ground-truth, x_{true} , is $e_{\text{mean}} \approx -3.84$ cm, demonstrating a small bias. This is significantly better than the MAP estimator, which had a bias of -33.0 cm on this same metric. The average squared error is approximately the same, with $e_{\text{sq}} \approx 4.32$ m².

Although it is difficult to show analytically, it is plausible to think that the iterated sigma-point method is trying to converge to the mean of the full posterior rather than the mode. If what we care about is

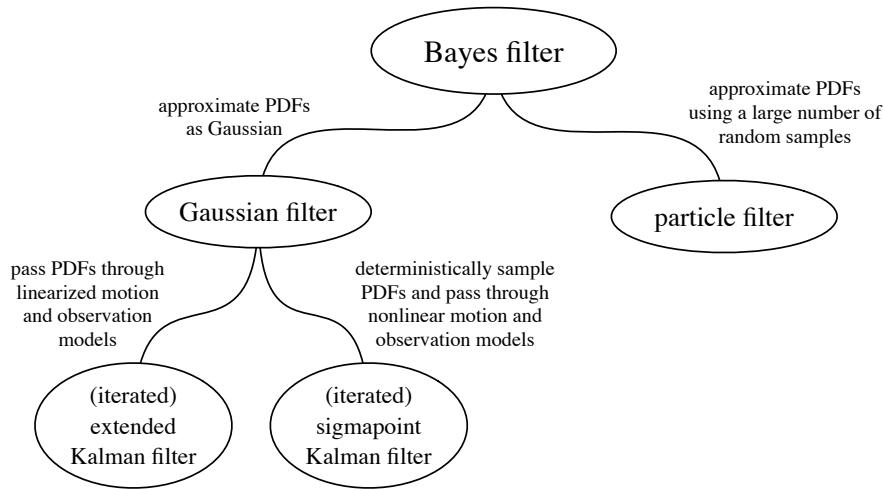


Figure 4.16
Taxonomy of the different filtering methods, showing their relationships to the Bayes filter.

matching up against groundtruth, matching the mean of the full posterior could be an interesting avenue.

4.2.12 Taxonomy of Filters

Figure 4.16 provides a summary of the methods we have discussed in this section on nonlinear recursive state estimation. We can think of each of the methods having a place in a larger taxonomy, with the Bayes filter at the top position. Depending on the approximations made, we wind up with the different filters discussed.

It is also worth recalling the role of iteration in our filter methods. Without iteration, both the EKF and SPKF were difficult to relate back to the full Bayesian posterior. However, we saw that the ‘mean’ of the IEKF converges to the MAP solution, while the mean of the ISPKF comes quite close to the mean of the full posterior. We will use these lessons in the next section on batch estimation, where we will attempt to estimate entire trajectories at once.

4.3 Batch Discrete-Time Estimation

In this section, we take a step back and question how valid the Bayes filter really is given the fact that we always have to implement it approximately and therefore are violating the Markov assumption on which it is predicated. We propose that a much better starting point in deriving nonlinear filters (i.e., better than the Bayes filter) is the nonlinear version of the batch estimator we first introduced in the chapter on linear-Gaussian estimation. Setting the estimation problem up as an optimization problem affords a different perspective that helps explain the shortcomings of all variants of the EKF.

4.3.1 Maximum A Posteriori

In this section, we revisit to our approach to linear-Gaussian estimation problems and batch optimization, and we introduce the Gauss-Newton method to solve our nonlinear version of this estimation problem. This optimization approach can be viewed as the MAP approach once again. We first set up our objective function that we seek to minimize, then consider methods to solve it.

Objective Function

We seek to construct an objective function that we will minimize with respect to

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_0 \\ \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_K \end{bmatrix}, \quad (4.108)$$

which represents the entire trajectory that we want to estimate.

Recall the linear-Gaussian objective function given by Equations (3.10) and (3.9). It took the form of a squared Mahalanobis distance and was proportional to the negative log likelihood of the state given all the data. For the nonlinear case, we define the errors with respect to the prior and measurements to be

$$\mathbf{e}_{v,k}(\mathbf{x}) = \begin{cases} \check{\mathbf{x}}_0 - \mathbf{x}_0, & k = 0 \\ \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{v}_k, \mathbf{0}) - \mathbf{x}_k, & k = 1 \dots K \end{cases}, \quad (4.109a)$$

$$\mathbf{e}_{y,k}(\mathbf{x}) = \mathbf{y}_k - \mathbf{g}(\mathbf{x}_k, \mathbf{0}), \quad k = 0 \dots K, \quad (4.109b)$$

so that the contributions to the objective function are

$$J_{v,k}(\mathbf{x}) = \frac{1}{2} \mathbf{e}_{v,k}(\mathbf{x})^T \mathbf{W}_{v,k}^{-1} \mathbf{e}_{v,k}(\mathbf{x}), \quad (4.110a)$$

$$J_{y,k}(\mathbf{x}) = \frac{1}{2} \mathbf{e}_{y,k}(\mathbf{x})^T \mathbf{W}_{y,k}^{-1} \mathbf{e}_{y,k}(\mathbf{x}). \quad (4.110b)$$

The overall objective function is then

$$J(\mathbf{x}) = \sum_{k=0}^K (J_{v,k}(\mathbf{x}) + J_{y,k}(\mathbf{x})). \quad (4.111)$$

Note that we can generally think of $\mathbf{W}_{v,k}$ and $\mathbf{W}_{y,k}$ simply as symmetric positive-definite matrix weights. By choosing these to be related to the covariances of the measurement noises, minimizing the objective function is equivalent to maximizing the joint likelihood of the state given all the data.

We further define

$$\mathbf{e}(\mathbf{x}) = \begin{bmatrix} \mathbf{e}_v(\mathbf{x}) \\ \mathbf{e}_y(\mathbf{x}) \end{bmatrix}, \quad \mathbf{e}_v(\mathbf{x}) = \begin{bmatrix} \mathbf{e}_{v,0}(\mathbf{x}) \\ \vdots \\ \mathbf{e}_{v,K}(\mathbf{x}) \end{bmatrix}, \quad \mathbf{e}_y(\mathbf{x}) = \begin{bmatrix} \mathbf{e}_{y,0}(\mathbf{x}) \\ \vdots \\ \mathbf{e}_{y,K}(\mathbf{x}) \end{bmatrix}, \quad (4.112a)$$

$$\mathbf{W} = \text{diag}(\mathbf{W}_v, \mathbf{W}_y), \quad \mathbf{W}_v = \text{diag}(\mathbf{W}_{v,0}, \dots, \mathbf{W}_{v,K}), \quad (4.112b)$$

$$\mathbf{W}_y = \text{diag}(\mathbf{W}_{y,0}, \dots, \mathbf{W}_{y,K}), \quad (4.112c)$$

so that the objective function can be written as

$$J(\mathbf{x}) = \frac{1}{2} \mathbf{e}(\mathbf{x})^T \mathbf{W}^{-1} \mathbf{e}(\mathbf{x}). \quad (4.113)$$

We can further define the modified error term,

$$\mathbf{u}(\mathbf{x}) = \mathbf{L} \mathbf{e}(\mathbf{x}), \quad (4.114)$$

where $\mathbf{L}^T \mathbf{L} = \mathbf{W}^{-1}$ (i.e., from a Cholesky decomposition since \mathbf{W} is symmetric positive-definite). Using these definitions, we can write the objective function simply as

$$J(\mathbf{x}) = \frac{1}{2} \mathbf{u}(\mathbf{x})^T \mathbf{u}(\mathbf{x}). \quad (4.115)$$

This is precisely in a quadratic form, but not with respect to the design variables, \mathbf{x} . Our goal is to determine the optimum design parameter, $\hat{\mathbf{x}}$, that minimizes the objective function:

$$\hat{\mathbf{x}} = \arg \min_{\mathbf{x}} J(\mathbf{x}). \quad (4.116)$$

We can apply many nonlinear optimization techniques to minimize this expression due to its quadratic nature. A typical technique to use is Gauss-Newton optimization, but there are many other possibilities. The more important issue is that we are considering this as a nonlinear optimization problem. We will derive the Gauss-Newton algorithm by way of Newton's method.

Newton's Method

Newton's method works by iteratively approximating the (differentiable) objective function by a quadratic function and then jumping to (or moving towards) the minimum. Suppose we have an initial guess, or operating point, for the design parameter, \mathbf{x}_{op} . We use a three-term Taylor-series expansion to approximate J as a quadratic function,

$$J(\mathbf{x}_{\text{op}} + \delta \mathbf{x}) \approx J(\mathbf{x}_{\text{op}}) + \underbrace{\left(\frac{\partial J(\mathbf{x})}{\partial \mathbf{x}} \Big|_{\mathbf{x}_{\text{op}}} \right)}_{\text{Jacobian}} \delta \mathbf{x} + \frac{1}{2} \delta \mathbf{x}^T \underbrace{\left(\frac{\partial^2 J(\mathbf{x})}{\partial \mathbf{x} \partial \mathbf{x}^T} \Big|_{\mathbf{x}_{\text{op}}} \right)}_{\text{Hessian}} \delta \mathbf{x}, \quad (4.117)$$

of $\delta\mathbf{x}$, a ‘small’ change to the initial guess, \mathbf{x}_{op} . We note that the symmetric Hessian matrix needs to be positive-definite for this method to work (otherwise there is no well-defined minimum to the quadratic approximation).

The next step is to find the value of $\delta\mathbf{x}$ that minimizes this quadratic approximation. We can do this by taking the derivative with respect to $\delta\mathbf{x}$ and setting to zero to find a critical point:

$$\begin{aligned} \frac{\partial J(\mathbf{x}_{op} + \delta\mathbf{x})}{\partial \delta\mathbf{x}} &= \left(\frac{\partial J(\mathbf{x})}{\partial \mathbf{x}} \Big|_{\mathbf{x}_{op}} \right) + \delta\mathbf{x}^{*T} \left(\frac{\partial^2 J(\mathbf{x})}{\partial \mathbf{x} \partial \mathbf{x}^T} \Big|_{\mathbf{x}_{op}} \right) = \mathbf{0} \\ \Rightarrow \quad \left(\frac{\partial^2 J(\mathbf{x})}{\partial \mathbf{x} \partial \mathbf{x}^T} \Big|_{\mathbf{x}_{op}} \right) \delta\mathbf{x}^* &= - \left(\frac{\partial J(\mathbf{x})}{\partial \mathbf{x}} \Big|_{\mathbf{x}_{op}} \right)^T. \end{aligned} \quad (4.118)$$

The last line is just a linear system of equations and can be solved when the Hessian is invertible (which it must be, since it was assumed to be positive-definite above). We may then update our operating point according to:

$$\mathbf{x}_{op} \leftarrow \mathbf{x}_{op} + \delta\mathbf{x}^*. \quad (4.119)$$

This procedure iterates until $\delta\mathbf{x}^*$ becomes sufficiently small. A few comments about Newton’s method:

- (i) It is ‘locally convergent’, which means the successive approximations are guaranteed to converge to a solution when the initial guess is already close enough to the solution. For a complex nonlinear objective function, this is really the best we can expect (i.e., global convergence is difficult to achieve).
- (ii) The rate of convergence is quadratic (i.e., it converges much faster than simple gradient descent).
- (iii) It can be difficult to implement because the Hessian must be computed.

The Gauss-Newton method approximates Newton’s method further, in the case of a special form of objective function.

Gauss-Newton Method

Let us now return to the nonlinear quadratic objective function we have in Equation (4.115). In this case, the Jacobian and Hessian matrices

are

$$\text{Jacobian: } \frac{\partial J(\mathbf{x})}{\partial \mathbf{x}} \Big|_{\mathbf{x}_{\text{op}}} = \mathbf{u}(\mathbf{x}_{\text{op}})^T \left(\frac{\partial \mathbf{u}(\mathbf{x})}{\partial \mathbf{x}} \Big|_{\mathbf{x}_{\text{op}}} \right), \quad (4.120a)$$

$$\begin{aligned} \text{Hessian: } \frac{\partial^2 J(\mathbf{x})}{\partial \mathbf{x} \partial \mathbf{x}^T} \Big|_{\mathbf{x}_{\text{op}}} &= \left(\frac{\partial \mathbf{u}(\mathbf{x})}{\partial \mathbf{x}} \Big|_{\mathbf{x}_{\text{op}}} \right)^T \left(\frac{\partial \mathbf{u}(\mathbf{x})}{\partial \mathbf{x}} \Big|_{\mathbf{x}_{\text{op}}} \right) \\ &\quad + \sum_{i=1}^M u_i(\mathbf{x}_{\text{op}}) \left(\frac{\partial^2 u_i(\mathbf{x})}{\partial \mathbf{x} \partial \mathbf{x}^T} \Big|_{\mathbf{x}_{\text{op}}} \right), \end{aligned} \quad (4.120b)$$

where $\mathbf{u}(\mathbf{x}) = (u_1(\mathbf{x}), \dots, u_i(\mathbf{x}), \dots, u_M(\mathbf{x}))$. We have so far not made any approximations.

Looking to the expression for the Hessian, we assert that near the minimum of J , the second term is small relative to the first. One intuition behind this is that near the optimum, we should have $u_i(\mathbf{x})$ small (and ideally zero). We thus approximate the Hessian according to

$$\frac{\partial^2 J(\mathbf{x})}{\partial \mathbf{x} \partial \mathbf{x}^T} \Big|_{\mathbf{x}_{\text{op}}} \approx \left(\frac{\partial \mathbf{u}(\mathbf{x})}{\partial \mathbf{x}} \Big|_{\mathbf{x}_{\text{op}}} \right)^T \left(\frac{\partial \mathbf{u}(\mathbf{x})}{\partial \mathbf{x}} \Big|_{\mathbf{x}_{\text{op}}} \right), \quad (4.121)$$

which does not involve any second derivatives. Substituting (4.120a) and (4.121) into the Newton update represented by (4.118), we have

$$\left(\frac{\partial \mathbf{u}(\mathbf{x})}{\partial \mathbf{x}} \Big|_{\mathbf{x}_{\text{op}}} \right)^T \left(\frac{\partial \mathbf{u}(\mathbf{x})}{\partial \mathbf{x}} \Big|_{\mathbf{x}_{\text{op}}} \right) \delta \mathbf{x}^* = - \left(\frac{\partial \mathbf{u}(\mathbf{x})}{\partial \mathbf{x}} \Big|_{\mathbf{x}_{\text{op}}} \right)^T \mathbf{u}(\mathbf{x}_{\text{op}}), \quad (4.122)$$

which is the classic Gauss-Newton update method. Again, this is iterated to convergence.

Gauss-Newton Method: Alternative Derivation

The other way to think about the Gauss-Newton method is to start with a Taylor expansion of $\mathbf{u}(\mathbf{x})$, instead of $J(\mathbf{x})$. The approximation in this case is

$$\mathbf{u}(\mathbf{x}_{\text{op}} + \delta \mathbf{x}) \approx \mathbf{u}(\mathbf{x}_{\text{op}}) + \left(\frac{\partial \mathbf{u}(\mathbf{x})}{\partial \mathbf{x}} \Big|_{\mathbf{x}_{\text{op}}} \right) \delta \mathbf{x}. \quad (4.123)$$

Substituting into J , we have

$$\begin{aligned} J(\mathbf{x}_{\text{op}} + \delta \mathbf{x}) &\approx \frac{1}{2} \left(\mathbf{u}(\mathbf{x}_{\text{op}}) + \left(\frac{\partial \mathbf{u}(\mathbf{x})}{\partial \mathbf{x}} \Big|_{\mathbf{x}_{\text{op}}} \right) \delta \mathbf{x} \right)^T \left(\mathbf{u}(\mathbf{x}_{\text{op}}) + \left(\frac{\partial \mathbf{u}(\mathbf{x})}{\partial \mathbf{x}} \Big|_{\mathbf{x}_{\text{op}}} \right) \delta \mathbf{x} \right). \end{aligned} \quad (4.124)$$

Minimizing with respect to $\delta\mathbf{x}$ gives

$$\begin{aligned} \frac{\partial J(\mathbf{x}_{\text{op}} + \delta\mathbf{x})}{\partial \delta\mathbf{x}} &= \left(\mathbf{u}(\mathbf{x}_{\text{op}}) + \left(\frac{\partial \mathbf{u}(\mathbf{x})}{\partial \mathbf{x}} \Big|_{\mathbf{x}_{\text{op}}} \right) \delta\mathbf{x}^* \right)^T \left(\frac{\partial \mathbf{u}(\mathbf{x})}{\partial \mathbf{x}} \Big|_{\mathbf{x}_{\text{op}}} \right) = \mathbf{0} \\ \Rightarrow \left(\frac{\partial \mathbf{u}(\mathbf{x})}{\partial \mathbf{x}} \Big|_{\mathbf{x}_{\text{op}}} \right)^T \left(\frac{\partial \mathbf{u}(\mathbf{x})}{\partial \mathbf{x}} \Big|_{\mathbf{x}_{\text{op}}} \right) \delta\mathbf{x}^* &= - \left(\frac{\partial \mathbf{u}(\mathbf{x})}{\partial \mathbf{x}} \Big|_{\mathbf{x}_{\text{op}}} \right)^T \mathbf{u}(\mathbf{x}_{\text{op}}), \end{aligned} \quad (4.125)$$

which is the same update as in (4.122). We will employ this shortcut to the Gauss-Newton method in a later chapter when confronted with dealing with nonlinearities in the form of rotations.

Practical Patches to Gauss-Newton

Since the Gauss-Newton method is not guaranteed to converge (owing to the approximate Hessian matrix), we can make two practical patches to help with convergence:

- (i) Once the optimal update is computed, $\delta\mathbf{x}^*$, we perform the actual update according to

$$\mathbf{x}_{\text{op}} \leftarrow \mathbf{x}_{\text{op}} + \alpha \delta\mathbf{x}^*, \quad (4.126)$$

where $\alpha \in [0, 1]$ is a user-definable parameter. Performing a line search for the best value of α works well in practice. This works because $\delta\mathbf{x}^*$ is a descent direction; we are just adjusting how far we step in this direction to be a bit more conservative towards robustness (rather than speed).

- (ii) We can use the Levenberg-Marquardt modification to the Gauss-Newton method:

$$\begin{aligned} \left(\left(\frac{\partial \mathbf{u}(\mathbf{x})}{\partial \mathbf{x}} \Big|_{\mathbf{x}_{\text{op}}} \right)^T \left(\frac{\partial \mathbf{u}(\mathbf{x})}{\partial \mathbf{x}} \Big|_{\mathbf{x}_{\text{op}}} \right) + \lambda \mathbf{D} \right) \delta\mathbf{x}^* \\ = - \left(\frac{\partial \mathbf{u}(\mathbf{x})}{\partial \mathbf{x}} \Big|_{\mathbf{x}_{\text{op}}} \right)^T \mathbf{u}(\mathbf{x}_{\text{op}}), \end{aligned} \quad (4.127)$$

where \mathbf{D} is a positive diagonal matrix. When $\mathbf{D} = \mathbf{1}$, we can see that as $\lambda \geq 0$ becomes very big, the Hessian is relatively small, and we have

$$\delta\mathbf{x}^* \approx - \frac{1}{\lambda} \left(\frac{\partial \mathbf{u}(\mathbf{x})}{\partial \mathbf{x}} \Big|_{\mathbf{x}_{\text{op}}} \right)^T \mathbf{u}(\mathbf{x}_{\text{op}}), \quad (4.128)$$

which corresponds to a very small step in the direction of steepest descent (i.e., the negative gradient). When $\lambda = 0$, we recover the usual Gauss-Newton update. The Levenberg-Marquardt method can work well in situations when the Hessian approximation is

poor or is poorly conditioned by slowly increasing λ to improve conditioning.

We can also combine both of these patches to give us the most options in controlling convergence.

Gauss-Newton Update in Terms of Errors

Recalling that

$$\mathbf{u}(\mathbf{x}) = \mathbf{L} \mathbf{e}(\mathbf{x}), \quad (4.129)$$

with \mathbf{L} a constant, we substitute this into the Gauss-Newton update to see that, in terms of the error, $\mathbf{e}(\mathbf{x})$, we have

$$(\mathbf{H}^T \mathbf{W}^{-1} \mathbf{H}) \delta \mathbf{x}^* = \mathbf{H}^T \mathbf{W}^{-1} \mathbf{e}(\mathbf{x}_{\text{op}}), \quad (4.130)$$

with

$$\mathbf{H} = - \left. \frac{\partial \mathbf{e}(\mathbf{x})}{\partial \mathbf{x}} \right|_{\mathbf{x}_{\text{op}}}, \quad (4.131)$$

and where we have used $\mathbf{L}^T \mathbf{L} = \mathbf{W}^{-1}$.

Another way to view this is to notice that

$$J(\mathbf{x}_{\text{op}} + \delta \mathbf{x}) \approx \frac{1}{2} (\mathbf{e}(\mathbf{x}_{\text{op}}) - \mathbf{H} \delta \mathbf{x})^T \mathbf{W}^{-1} (\mathbf{e}(\mathbf{x}_{\text{op}}) - \mathbf{H} \delta \mathbf{x}), \quad (4.132)$$

where $\mathbf{e}(\mathbf{x}_{\text{op}}) = \mathbf{L}^{-1} \mathbf{u}(\mathbf{x}_{\text{op}})$, is the quadratic approximation of the objective function in terms of the error. Minimizing this with respect to $\delta \mathbf{x}$ yields the Gauss-Newton update.

Batch Estimation

We now return to our specific estimation problem and apply the Gauss-Newton optimization method. We will use the ‘shortcut’ approach and thus begin by approximating our error expressions:

$$\mathbf{e}_{v,k}(\mathbf{x}_{\text{op}} + \delta \mathbf{x}) \approx \begin{cases} \mathbf{e}_{v,0}(\mathbf{x}_{\text{op}}) - \delta \mathbf{x}_0, & k = 0 \\ \mathbf{e}_{v,k}(\mathbf{x}_{\text{op}}) + \mathbf{F}_{k-1} \delta \mathbf{x}_{k-1} - \delta \mathbf{x}_k, & k = 1 \dots K \end{cases}, \quad (4.133)$$

$$\mathbf{e}_{y,k}(\mathbf{x}_{\text{op}} + \delta \mathbf{x}) \approx \mathbf{e}_{y,k}(\mathbf{x}_{\text{op}}) - \mathbf{G}_k \delta \mathbf{x}_k, \quad k = 0 \dots K, \quad (4.134)$$

where

$$\mathbf{e}_{v,k}(\mathbf{x}_{\text{op}}) \approx \begin{cases} \check{\mathbf{x}}_0 - \mathbf{x}_{\text{op},0}, & k = 0 \\ \mathbf{f}(\mathbf{x}_{\text{op},k-1}, \mathbf{v}_k, \mathbf{0}) - \mathbf{x}_{\text{op},k}, & k = 1 \dots K \end{cases}, \quad (4.135)$$

$$\mathbf{e}_{y,k}(\mathbf{x}_{\text{op}}) \approx \mathbf{y}_k - \mathbf{g}(\mathbf{x}_{\text{op},k}, \mathbf{0}), \quad k = 0 \dots K, \quad (4.136)$$

and we require definitions of the Jacobians of the nonlinear motion and observations models given by

$$\mathbf{F}_{k-1} = \left. \frac{\partial \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{v}_k, \mathbf{w}_k)}{\partial \mathbf{x}_{k-1}} \right|_{\mathbf{x}_{\text{op},k-1}, \mathbf{v}_k, \mathbf{0}}, \quad \mathbf{G}_k = \left. \frac{\partial \mathbf{g}(\mathbf{x}_k, \mathbf{n}_k)}{\partial \mathbf{x}_k} \right|_{\mathbf{x}_{\text{op},k}, \mathbf{0}}. \quad (4.137)$$

Then, if we let the matrix weights be given by

$$\mathbf{W}_{v,k} = \mathbf{Q}'_k, \quad \mathbf{W}_{y,k} = \mathbf{R}'_k, \quad (4.138)$$

we can define

$$\delta\mathbf{x} = \begin{bmatrix} \delta\mathbf{x}_0 \\ \delta\mathbf{x}_1 \\ \delta\mathbf{x}_2 \\ \vdots \\ \delta\mathbf{x}_K \end{bmatrix}, \quad \mathbf{H} = \begin{bmatrix} \mathbf{1} & & & & \\ -\mathbf{F}_0 & \mathbf{1} & & & \\ & -\mathbf{F}_1 & \ddots & & \\ & & \ddots & \mathbf{1} & \\ & & & -\mathbf{F}_{K-1} & \mathbf{1} \\ \hline \mathbf{G}_0 & & & & \\ & \mathbf{G}_1 & & & \\ & & \mathbf{G}_2 & & \\ & & & \ddots & \\ & & & & \mathbf{G}_K \end{bmatrix}, \quad (4.139a)$$

$$\mathbf{e}(\mathbf{x}_{\text{op}}) = \begin{bmatrix} \mathbf{e}_{v,0}(\mathbf{x}_{\text{op}}) \\ \mathbf{e}_{v,1}(\mathbf{x}_{\text{op}}) \\ \vdots \\ \frac{\mathbf{e}_{v,K}(\mathbf{x}_{\text{op}})}{\mathbf{e}_{y,0}(\mathbf{x}_{\text{op}})} \\ \mathbf{e}_{y,1}(\mathbf{x}_{\text{op}}) \\ \vdots \\ \mathbf{e}_{y,K}(\mathbf{x}_{\text{op}}) \end{bmatrix}, \quad (4.139b)$$

and

$$\mathbf{W} = \text{diag} (\check{\mathbf{P}}_0, \mathbf{Q}'_1, \dots, \mathbf{Q}'_K, \mathbf{R}'_0, \mathbf{R}'_1, \dots, \mathbf{R}'_K), \quad (4.140)$$

which are identical in structure to the matrices in the linear batch case, summarized in (3.3.1), with a few extra subscripts to show time dependence as well as the Jacobians of the motion/observation models with respect to the noise variables. Under these definitions, our Gauss-Newton update is given by

$$\underbrace{(\mathbf{H}^T \mathbf{W}^{-1} \mathbf{H})}_{\text{block-tridiagonal}} \delta\mathbf{x}^* = \mathbf{H}^T \mathbf{W}^{-1} \mathbf{e}(\mathbf{x}_{\text{op}}). \quad (4.141)$$

This is very comparable to the linear-Gaussian batch case. The key difference to remember here is that we are in fact iterating our solution for the entire trajectory, \mathbf{x} . We could at this point recover the recursive EKF from our batch solution using similar logic to the linear-Gaussian case.

4.3.2 Bayesian Inference

We can also get to the same batch update equations from a Bayesian-inference perspective. Assume we begin with an initial guess for the entire trajectory, \mathbf{x}_{op} . We can linearize the motion model about this guess and construct a prior over the whole trajectory using all the inputs. The linearized motion model is

$$\mathbf{x}_k \approx \mathbf{f}(\mathbf{x}_{\text{op},k-1}, \mathbf{v}_k, \mathbf{0}) + \mathbf{F}_{k-1} (\mathbf{x}_{k-1} - \mathbf{x}_{\text{op},k-1}) + \mathbf{w}'_k, \quad (4.142)$$

where the Jacobian, \mathbf{F}_{k-1} , is the same as the previous section. After a bit of manipulation, we can write this in lifted form as

$$\mathbf{x} = \mathbf{F}(\boldsymbol{\nu} + \mathbf{w}'), \quad (4.143)$$

where

$$\boldsymbol{\nu} = \begin{bmatrix} \check{\mathbf{x}}_0 \\ \mathbf{f}(\mathbf{x}_{\text{op},0}, \mathbf{v}_1, \mathbf{0}) - \mathbf{F}_0 \mathbf{x}_{\text{op},0} \\ \mathbf{f}(\mathbf{x}_{\text{op},1}, \mathbf{v}_2, \mathbf{0}) - \mathbf{F}_1 \mathbf{x}_{\text{op},1} \\ \vdots \\ \mathbf{f}(\mathbf{x}_{\text{op},K-1}, \mathbf{v}_K, \mathbf{0}) - \mathbf{F}_{K-1} \mathbf{x}_{\text{op},K-1} \end{bmatrix}, \quad (4.144a)$$

$$\mathbf{F} = \begin{bmatrix} 1 & & & & & \\ \mathbf{F}_0 & 1 & & & & \\ \mathbf{F}_1 \mathbf{F}_0 & \mathbf{F}_1 & 1 & & & \\ \vdots & \vdots & \vdots & \ddots & & \\ \mathbf{F}_{K-2} \cdots \mathbf{F}_0 & \mathbf{F}_{K-2} \cdots \mathbf{F}_1 & \mathbf{F}_{K-2} \cdots \mathbf{F}_2 & \cdots & 1 & \\ \mathbf{F}_{K-1} \cdots \mathbf{F}_0 & \mathbf{F}_{K-1} \cdots \mathbf{F}_1 & \mathbf{F}_{K-1} \cdots \mathbf{F}_2 & \cdots & \mathbf{F}_{K-1} & 1 \end{bmatrix}, \quad (4.144b)$$

$$\mathbf{Q}' = \text{diag}(\check{\mathbf{P}}_0, \mathbf{Q}'_1, \mathbf{Q}'_2, \dots, \mathbf{Q}'_K), \quad (4.144c)$$

and $\mathbf{w}' \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}')$. For the mean of the prior, $\check{\mathbf{x}}$, we then simply have

$$\check{\mathbf{x}} = E[\mathbf{x}] = E[\mathbf{F}(\boldsymbol{\nu} + \mathbf{w}')] = \mathbf{F}\boldsymbol{\nu}. \quad (4.145)$$

For the covariance of the prior, $\check{\mathbf{P}}$, we have

$$\check{\mathbf{P}} = E[(\mathbf{x} - E[\mathbf{x}])(\mathbf{x} - E[\mathbf{x}])^T] = \mathbf{F} E[\mathbf{w}' \mathbf{w}'^T] \mathbf{F}^T = \mathbf{F} \mathbf{Q}' \mathbf{F}^T. \quad (4.146)$$

Thus, the prior can be summarized as $\mathbf{x} \sim \mathcal{N}(\mathbf{F}\boldsymbol{\nu}, \mathbf{F}\mathbf{Q}'\mathbf{F}^T)$. The $(\cdot)'$ notation is used to indicate that the Jacobian with respect to the noise is incorporated into the quantity.

The linearized observation model is

$$\mathbf{y}_k \approx \mathbf{g}(\mathbf{x}_{\text{op},k}, \mathbf{0}) + \mathbf{G}_k (\mathbf{x}_{k-1} - \mathbf{x}_{\text{op},k-1}) + \mathbf{n}'_k, \quad (4.147)$$

which can be written in lifted form as

$$\mathbf{y} = \mathbf{y}_{\text{op}} + \mathbf{G}(\mathbf{x} - \mathbf{x}_{\text{op}}) + \mathbf{n}', \quad (4.148)$$

where

$$\mathbf{y}_{\text{op}} = \begin{bmatrix} \mathbf{g}(\mathbf{x}_{\text{op},0}, \mathbf{0}) \\ \mathbf{g}(\mathbf{x}_{\text{op},1}, \mathbf{0}) \\ \vdots \\ \mathbf{g}(\mathbf{x}_{\text{op},K}, \mathbf{0}) \end{bmatrix}, \quad (4.149\text{a})$$

$$\mathbf{G} = \text{diag}(\mathbf{G}_0, \mathbf{G}_1, \mathbf{G}_2, \dots, \mathbf{G}_K), \quad (4.149\text{b})$$

$$\mathbf{R} = \text{diag}(\mathbf{R}'_0, \mathbf{R}'_1, \mathbf{R}'_2, \dots, \mathbf{R}'_K), \quad (4.149\text{c})$$

and $\mathbf{n}' \sim \mathcal{N}(\mathbf{0}, \mathbf{R}')$. It is fairly easy to see that

$$E[\mathbf{y}] = \mathbf{y}_{\text{op}} + \mathbf{G}(\check{\mathbf{x}} - \mathbf{x}_{\text{op}}), \quad (4.150\text{a})$$

$$E[(\mathbf{y} - E[\mathbf{y}])(\mathbf{y} - E[\mathbf{y}])^T] = \check{\mathbf{P}}\mathbf{G}\mathbf{G}^T + \mathbf{R}', \quad (4.150\text{b})$$

$$E[(\mathbf{y} - E[\mathbf{y}])(\mathbf{x} - E[\mathbf{x}])^T] = \check{\mathbf{P}}\mathbf{G}. \quad (4.150\text{c})$$

Again, the $(\cdot)'$ notation is used to indicate that the Jacobian with respect to the noise is incorporated into the quantity.

With these quantities in hand, we can write a joint density for the lifted trajectory and measurements as

$$p(\mathbf{x}, \mathbf{y} | \boldsymbol{\nu}) = \mathcal{N}\left(\begin{bmatrix} \check{\mathbf{x}} \\ \mathbf{y}_{\text{op}} + \mathbf{G}(\check{\mathbf{x}} - \mathbf{x}_{\text{op}}) \end{bmatrix}, \begin{bmatrix} \check{\mathbf{P}} & \check{\mathbf{P}}\mathbf{G}^T \\ \mathbf{G}\check{\mathbf{P}} & \mathbf{G}\check{\mathbf{P}}\mathbf{G}^T + \mathbf{R}' \end{bmatrix}\right), \quad (4.151)$$

which is quite similar to the expression for the IEKF situation in (4.42), but now for the whole trajectory rather than just one timestep. Using the usual relationship from (2.53b), we can immediately write the Gaussian posterior as

$$p(\mathbf{x} | \boldsymbol{\nu}, \mathbf{y}) = \mathcal{N}\left(\hat{\mathbf{x}}, \hat{\mathbf{P}}\right), \quad (4.152)$$

where

$$\mathbf{K} = \check{\mathbf{P}}\mathbf{G}^T (\mathbf{G}\check{\mathbf{P}}\mathbf{G}^T + \mathbf{R}')^{-1}, \quad (4.153\text{a})$$

$$\hat{\mathbf{P}} = (\mathbf{I} - \mathbf{K}\mathbf{G})\check{\mathbf{P}}, \quad (4.153\text{b})$$

$$\hat{\mathbf{x}} = \check{\mathbf{x}} + \mathbf{K}(\mathbf{y} - \mathbf{y}_{\text{op}} - \mathbf{G}(\check{\mathbf{x}} - \mathbf{x}_{\text{op}})). \quad (4.153\text{c})$$

Using the SMW identity from (2.75), we can rearrange the equation for the posterior mean to be

$$\left(\check{\mathbf{P}}^{-1} + \mathbf{G}^T \mathbf{R}'^{-1} \mathbf{G}\right) \delta\mathbf{x}^* = \check{\mathbf{P}}^{-1}(\check{\mathbf{x}} - \mathbf{x}_{\text{op}}) + \mathbf{G}^T \mathbf{R}'^{-1}(\mathbf{y} - \mathbf{y}_{\text{op}}), \quad (4.154)$$

where $\delta\mathbf{x}^* = \hat{\mathbf{x}} - \mathbf{x}_{\text{op}}$. Inserting the details of the prior, this becomes

$$\underbrace{\left(\mathbf{F}^{-T} \mathbf{Q}'^{-1} \mathbf{F}^{-1} + \mathbf{G}^T \mathbf{R}'^{-1} \mathbf{G}\right)}_{\text{block-tridiagonal}} \delta\mathbf{x}^* = \mathbf{F}^{-T} \mathbf{Q}'^{-1} (\boldsymbol{\nu} - \mathbf{F}^{-1} \mathbf{x}_{\text{op}}) + \mathbf{G}^T \mathbf{R}'^{-1}(\mathbf{y} - \mathbf{y}_{\text{op}}). \quad (4.155)$$

Then, under the definitions

$$\mathbf{H} = \begin{bmatrix} \mathbf{F}^{-1} \\ \mathbf{G} \end{bmatrix}, \quad \mathbf{W} = \text{diag}(\mathbf{Q}', \mathbf{R}'), \quad \mathbf{e}(\mathbf{x}_{\text{op}}) = \begin{bmatrix} \boldsymbol{\nu} - \mathbf{F}^{-1}\mathbf{x}_{\text{op}} \\ \mathbf{y} - \mathbf{y}_{\text{op}} \end{bmatrix}, \quad (4.156)$$

we can rewrite this as

$$\underbrace{(\mathbf{H}^T \mathbf{W}^{-1} \mathbf{H})}_{\text{block-tridiagonal}} \delta \mathbf{x}^* = \mathbf{H}^T \mathbf{W}^{-1} \mathbf{e}(\mathbf{x}_{\text{op}}), \quad (4.157)$$

which is identical to the update equation from the previous section. As usual, we iterate to convergence. The difference between the Bayesian and MAP approaches basically comes down to on which side of the SMW identity one begins; plus, the Bayesian approach produces a covariance explicitly, although we have shown that the same thing can be extracted from the MAP approach. Note that it was our choice to iteratively relinearize about the mean of the best estimate so far, which caused the Bayesian approach to have the same ‘mean’ as the MAP solution. We saw this phenomenon previously in the IEKF section. We could also imagine making different choices than linearization in the batch case (e.g., particles, sigma points) to compute the required moments for the update equations, but we will not explore these possibilities here.

4.3.3 Maximum Likelihood

In this section, we consider a simplified version of our batch estimation problem, where we throw away the prior and use only the measurements for our solution.

Maximum Likelihood via Gauss-Newton

We will assume the observation model takes on a simplified form in which the measurement noise is purely additive (i.e., outside the non-linearity):

$$\mathbf{y}_k = \mathbf{g}_k(\mathbf{x}) + \mathbf{n}_k, \quad (4.158)$$

where $\mathbf{n}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k)$. Note that in this case, we allow for the possibility that the measurement function is changing with k and that it could depend on an arbitrary portion of the state, \mathbf{x} . We need no longer think of k as a time index, simply as a measurement index.

Without the prior, our objective function takes the form

$$J(\mathbf{x}) = \frac{1}{2} \sum_k (\mathbf{y}_k - \mathbf{g}_k(\mathbf{x}))^T \mathbf{R}_k^{-1} (\mathbf{y}_k - \mathbf{g}_k(\mathbf{x})) = -\log p(\mathbf{y}|\mathbf{x}) + C, \quad (4.159)$$

where C is a constant. Without the prior term in the objective function,

we refer to this as a *maximum likelihood (ML)* problem because finding the solution that minimizes the objective function also maximizes the likelihood of the measurements¹⁰:

$$\hat{\mathbf{x}} = \arg \min_{\mathbf{x}} J(\mathbf{x}) = \arg \max_{\mathbf{x}} p(\mathbf{y}|\mathbf{x}). \quad (4.160)$$

We can still use the Gauss-Newton algorithm to solve the ML problem, just as in the MAP case. We begin with an initial guess for the solution, \mathbf{x}_{op} . We then compute an optimal update, $\delta\mathbf{x}^*$, by solving

$$\left(\sum_k \mathbf{G}_k(\mathbf{x}_{\text{op}})^T \mathbf{R}_k^{-1} \mathbf{G}_k(\mathbf{x}_{\text{op}}) \right) \delta\mathbf{x}^* = \sum_k \mathbf{G}_k(\mathbf{x}_{\text{op}})^T \mathbf{R}_k^{-1} (\mathbf{y}_k - \mathbf{g}_k(\mathbf{x}_{\text{op}})), \quad (4.161)$$

where

$$\mathbf{G}_k(\mathbf{x}) = \frac{\partial \mathbf{g}_k(\mathbf{x})}{\partial \mathbf{x}} \quad (4.162)$$

is the Jacobian of the observation model with respect to the state. Finally, we apply the optimal update to our guess,

$$\mathbf{x}_{\text{op}} \leftarrow \mathbf{x}_{\text{op}} + \delta\mathbf{x}^*, \quad (4.163)$$

and iterate to convergence. Once converged, we take $\hat{\mathbf{x}} = \mathbf{x}_{\text{op}}$ as our estimate. At convergence, we should have that

$$\left. \frac{\partial J(\mathbf{x})}{\partial \mathbf{x}^T} \right|_{\hat{\mathbf{x}}} = - \sum_k \mathbf{G}_k(\hat{\mathbf{x}})^T \mathbf{R}_k^{-1} (\mathbf{y}_k - \mathbf{g}_k(\hat{\mathbf{x}})) = \mathbf{0} \quad (4.164)$$

for a minimum.

We will come back to this ML setup when discussing a problem called *bundle adjustment* in the later chapters.

Maximum Likelihood Bias Estimation

We have already seen in the simple example at the start of this chapter that the MAP method is biased with respect to average mean error. It turns out that the ML method is biased as well (unless the measurement model is linear). A classic paper by Box (1971) derives an approximate expression for the bias in ML, and we use this section to present it.

We will see below that we need a second-order Taylor expansion of $\mathbf{g}(\mathbf{x})$ while only a first-order expansion of $\mathbf{G}(\mathbf{x})$. Thus, we have the following approximate expressions:

$$\mathbf{g}_k(\hat{\mathbf{x}}) = \mathbf{g}_k(\mathbf{x} + \delta\mathbf{x}) \approx \mathbf{g}_k(\mathbf{x}) + \mathbf{G}_k(\mathbf{x}) \delta\mathbf{x} + \frac{1}{2} \sum_j \mathbf{1}_j \delta\mathbf{x}^T \mathbf{G}_{jk}(\mathbf{x}) \delta\mathbf{x}, \quad (4.165a)$$

$$\mathbf{G}_k(\hat{\mathbf{x}}) = \mathbf{G}_k(\mathbf{x} + \delta\mathbf{x}) \approx \mathbf{G}_k(\mathbf{x}) + \sum_j \mathbf{1}_j \delta\mathbf{x}^T \mathbf{G}_{jk}(\mathbf{x}), \quad (4.165b)$$

¹⁰ This is because the logarithm is a monotonically increasing function. Another way of looking at ML is that it is MAP with a uniform prior over all possible solutions.

where

$$\mathbf{g}_k(\mathbf{x}) = [g_{jk}(\mathbf{x})]_j, \quad \mathbf{G}_k(\mathbf{x}) = \frac{\partial \mathbf{g}_k(\mathbf{x})}{\partial \mathbf{x}}, \quad \mathcal{G}_{jk} = \frac{\partial g_{jk}(\mathbf{x})}{\partial \mathbf{x} \partial \mathbf{x}^T}, \quad (4.166)$$

and $\mathbf{1}_j$ is the j th column of the identity matrix. We have indicated whether each Jacobian/Hessian is evaluated at \mathbf{x} (the true state) or $\hat{\mathbf{x}}$ (our estimate). In this section, the quantity $\delta\mathbf{x} = \hat{\mathbf{x}} - \mathbf{x}$ will be the difference between our estimate and the true state on a given trial. Each time we change the measurement noise, we will get a different value for the estimate and hence $\delta\mathbf{x}$. We will seek an expression for the expected value of this difference, $E[\delta\mathbf{x}]$, over all possible realizations of the measurement noise; this represents the systematic error or *bias*.

As discussed above, after convergence of Gauss-Newton, the estimate, $\hat{\mathbf{x}}$, will satisfy the following optimality criterion:

$$\sum_k \mathbf{G}_k(\hat{\mathbf{x}})^T \mathbf{R}_k^{-1} (\mathbf{y}_k - \mathbf{g}_k(\hat{\mathbf{x}})) = \mathbf{0}, \quad (4.167)$$

or

$$\begin{aligned} & \sum_k \left(\mathbf{G}_k(\mathbf{x}) + \sum_j \mathbf{1}_j \delta\mathbf{x}^T \mathcal{G}_{jk}(\mathbf{x}) \right)^T \mathbf{R}_k^{-1} \\ & \times \left(\underbrace{\mathbf{y}_k - \mathbf{g}_k(\mathbf{x})}_{\mathbf{n}_k} - \mathbf{G}_k(\mathbf{x}) \delta\mathbf{x} - \frac{1}{2} \sum_j \mathbf{1}_j \delta\mathbf{x}^T \mathcal{G}_{jk}(\mathbf{x}) \delta\mathbf{x} \right) \approx \mathbf{0}, \end{aligned} \quad (4.168)$$

after substituting (4.165a) and (4.165b). We will assume that $\delta\mathbf{x}$ has up to quadratic dependence¹¹ on the stacked noise variable, $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$:

$$\delta\mathbf{x} = \mathbf{A}(\mathbf{x}) \mathbf{n} + \mathbf{b}(\mathbf{n}), \quad (4.169)$$

where $\mathbf{A}(\mathbf{x})$ is an unknown coefficient matrix and $\mathbf{b}(\mathbf{n})$ is an unknown quadratic function of \mathbf{n} . We will use \mathbf{P}_k , a projection matrix, to extract the k th noise variable from the stacked version: $\mathbf{n}_k = \mathbf{P}_k \mathbf{n}$. Substituting (4.169), we have that

$$\begin{aligned} & \sum_k \left(\mathbf{G}_k(\mathbf{x}) + \sum_j \mathbf{1}_j (\mathbf{A}(\mathbf{x}) \mathbf{n} + \mathbf{b}(\mathbf{n}))^T \mathcal{G}_{jk}(\mathbf{x}) \right)^T \\ & \times \mathbf{R}_k^{-1} \left(\mathbf{P}_k \mathbf{n} - \mathbf{G}_k(\mathbf{x}) (\mathbf{A}(\mathbf{x}) \mathbf{n} + \mathbf{b}(\mathbf{n})) \right. \\ & \left. - \frac{1}{2} \sum_j \mathbf{1}_j (\mathbf{A}(\mathbf{x}) \mathbf{n} + \mathbf{b}(\mathbf{n}))^T \mathcal{G}_{jk}(\mathbf{x}) (\mathbf{A}(\mathbf{x}) \mathbf{n} + \mathbf{b}(\mathbf{n})) \right) \approx \mathbf{0}. \end{aligned} \quad (4.170)$$

¹¹ In reality, there are an infinite number of terms, so this expression is a big approximation but may work for mildly nonlinear observation models.

Multiplying out and keeping terms up to quadratic in \mathbf{n} , we have

$$\begin{aligned}
 & \underbrace{\sum_k \mathbf{G}_k(\mathbf{x})^T \mathbf{R}_k^{-1} (\mathbf{P}_k - \mathbf{G}_k(\mathbf{x})\mathbf{A}(\mathbf{x})) \mathbf{n}}_{\mathbf{L}\mathbf{n} \text{ (linear in } \mathbf{n})} \\
 & + \underbrace{\sum_k \mathbf{G}_k(\mathbf{x})^T \mathbf{R}_k^{-1} \left(-\mathbf{G}_k(\mathbf{x})\mathbf{b}(\mathbf{n}) - \frac{1}{2} \sum_j \mathbf{1}_j^T \underbrace{\mathbf{n}^T \mathbf{A}(\mathbf{x})^T \mathcal{G}_{jk}(\mathbf{x}) \mathbf{A}(\mathbf{x}) \mathbf{n}}_{\text{scalar}} \right)}_{\mathbf{q}_1(\mathbf{n}) \text{ (quadratic in } \mathbf{n})} \\
 & + \underbrace{\sum_{j,k} \mathcal{G}_{jk}(\mathbf{x})^T \mathbf{A}(\mathbf{x}) \mathbf{n} \underbrace{\mathbf{1}_j^T \mathbf{R}_k^{-1} (\mathbf{P}_k - \mathbf{G}_k(\mathbf{x})\mathbf{A}(\mathbf{x})) \mathbf{n}}_{\text{scalar}}}_{\mathbf{q}_2(\mathbf{n}) \text{ (quadratic in } \mathbf{n})} \approx \mathbf{0}. \quad (4.171)
 \end{aligned}$$

To make the expression identically zero (up to second order in \mathbf{n}),

$$\mathbf{L}\mathbf{n} + \mathbf{q}_1(\mathbf{n}) + \mathbf{q}_2(\mathbf{n}) = \mathbf{0}, \quad (4.172)$$

we must have $\mathbf{L} = \mathbf{0}$. This follows by considering the case of the opposing sign of \mathbf{n} ,

$$-\mathbf{L}\mathbf{n} + \mathbf{q}_1(-\mathbf{n}) + \mathbf{q}_2(-\mathbf{n}) = \mathbf{0}, \quad (4.173)$$

and then noting that $\mathbf{q}_1(-\mathbf{n}) = \mathbf{q}_1(\mathbf{n})$ and $\mathbf{q}_2(-\mathbf{n}) = \mathbf{q}_2(\mathbf{n})$ owing to the quadratic nature of these terms. Subtracting the second case from the first, we have $2\mathbf{L}\mathbf{n} = \mathbf{0}$, and since \mathbf{n} can take on any value, it follows that $\mathbf{L} = \mathbf{0}$ and thus

$$\mathbf{A}(\mathbf{x}) = \mathbf{W}(\mathbf{x})^{-1} \sum_k \mathbf{G}_k(\mathbf{x})^T \mathbf{R}_k^{-1} \mathbf{P}_k, \quad (4.174)$$

where

$$\mathbf{W}(\mathbf{x}) = \sum_k \mathbf{G}_k(\mathbf{x})^T \mathbf{R}_k^{-1} \mathbf{G}_k(\mathbf{x}). \quad (4.175)$$

Choosing this value for $\mathbf{A}(\mathbf{x})$ and taking the expectation (over all values of \mathbf{n}), we are left with

$$E[\mathbf{q}_1(\mathbf{n})] + E[\mathbf{q}_2(\mathbf{n})] = \mathbf{0}. \quad (4.176)$$

Fortunately, it turns out that $E[\mathbf{q}_2(\mathbf{n})] = \mathbf{0}$. To see this, we need two identities:

$$\mathbf{A}(\mathbf{x})\mathbf{R}\mathbf{A}(\mathbf{x})^T \equiv \mathbf{W}(\mathbf{x})^{-1}, \quad (4.177a)$$

$$\mathbf{A}(\mathbf{x})\mathbf{R}\mathbf{P}_k^T \equiv \mathbf{W}(\mathbf{x})^{-1} \mathbf{G}_k(\mathbf{x})^T. \quad (4.177b)$$

The proofs of these are left to the reader. We then have

$$\begin{aligned}
E[\mathbf{q}_2(\mathbf{n})] &= E \left[\sum_{j,k} \mathcal{G}_{jk}(\mathbf{x})^T \mathbf{A}(\mathbf{x}) \mathbf{n} \mathbf{1}_j^T \mathbf{R}_k^{-1} (\mathbf{P}_k - \mathbf{G}_k(\mathbf{x}) \mathbf{A}(\mathbf{x})) \mathbf{n} \right] \\
&= \sum_{j,k} \mathcal{G}_{jk}(\mathbf{x})^T \mathbf{A}(\mathbf{x}) \underbrace{E[\mathbf{n} \mathbf{n}^T]}_{\mathbf{R}} (\mathbf{P}_k^T - \mathbf{A}(\mathbf{x})^T \mathbf{G}_k(\mathbf{x})^T) \mathbf{R}_k^{-1} \mathbf{1}_j \\
&= \sum_{j,k} \mathcal{G}_{jk}(\mathbf{x})^T \left(\underbrace{\mathbf{A}(\mathbf{x}) \mathbf{R} \mathbf{P}_k^T}_{\mathbf{W}(\mathbf{x})^{-1} \mathbf{G}_k(\mathbf{x})^T} - \underbrace{\mathbf{A}(\mathbf{x}) \mathbf{R} \mathbf{A}(\mathbf{x})^T}_{\mathbf{W}(\mathbf{x})^{-1}} \mathbf{G}_k(\mathbf{x})^T \right) \mathbf{R}_k^{-1} \mathbf{1}_j \\
&= \mathbf{0}, \tag{4.178}
\end{aligned}$$

where we have employed the above identities. We are thus left with

$$E[\mathbf{q}_1(\mathbf{n})] = \mathbf{0}, \tag{4.179}$$

or

$$\begin{aligned}
E[\mathbf{b}(\mathbf{n})] &= -\frac{1}{2} \mathbf{W}(\mathbf{x})^{-1} \sum_k \mathbf{G}_k(\mathbf{x})^T \mathbf{R}_k^{-1} \sum_j \mathbf{1}_j E[\mathbf{n}^T \mathbf{A}(\mathbf{x})^T \mathcal{G}_{jk}(\mathbf{x}) \mathbf{A}(\mathbf{x}) \mathbf{n}] \\
&= -\frac{1}{2} \mathbf{W}(\mathbf{x})^{-1} \sum_k \mathbf{G}_k(\mathbf{x})^T \mathbf{R}_k^{-1} \sum_j \mathbf{1}_j E[\text{tr}(\mathcal{G}_{jk}(\mathbf{x}) \mathbf{A}(\mathbf{x}) \mathbf{n} \mathbf{n}^T \mathbf{A}(\mathbf{x})^T)] \\
&= -\frac{1}{2} \mathbf{W}(\mathbf{x})^{-1} \sum_k \mathbf{G}_k(\mathbf{x})^T \mathbf{R}_k^{-1} \sum_j \mathbf{1}_j \text{tr}(\mathcal{G}_{jk}(\mathbf{x}) \mathbf{A}(\mathbf{x}) \underbrace{E[\mathbf{n} \mathbf{n}^T]}_{\mathbf{R}} \mathbf{A}(\mathbf{x})^T) \\
&= -\frac{1}{2} \mathbf{W}(\mathbf{x})^{-1} \sum_k \mathbf{G}_k(\mathbf{x})^T \mathbf{R}_k^{-1} \sum_j \mathbf{1}_j \text{tr}(\mathcal{G}_{jk}(\mathbf{x}) \mathbf{W}(\mathbf{x})^{-1}), \tag{4.180}
\end{aligned}$$

where $\text{tr}(\cdot)$ indicates the trace of a matrix. Looking back to (4.169), we see that

$$E[\delta \mathbf{x}] = \mathbf{A}(\mathbf{x}) \underbrace{E[\mathbf{n}]}_{\mathbf{0}} + E[\mathbf{b}(\mathbf{n})], \tag{4.181}$$

and so our final expression for the systematic part of the bias is

$$E[\delta \mathbf{x}] = -\frac{1}{2} \mathbf{W}(\mathbf{x})^{-1} \sum_k \mathbf{G}_k(\mathbf{x})^T \mathbf{R}_k^{-1} \sum_j \mathbf{1}_j \text{tr}(\mathcal{G}_{jk}(\mathbf{x}) \mathbf{W}(\mathbf{x})^{-1}). \tag{4.182}$$

To use this expression in operation, we will need to substitute our estimate, $\hat{\mathbf{x}}$, in place of \mathbf{x} when computing (4.182). Then we can update our estimate according to

$$\hat{\mathbf{x}} \leftarrow \hat{\mathbf{x}} - E[\delta \mathbf{x}], \tag{4.183}$$

to subtract off the bias. Note that this expression is only approximate and may only work well in mildly nonlinear situations.

4.3.4 Discussion

If we think of the EKF as an approximation of the full nonlinear Gauss-Newton (or even Newton) method applied to our estimation problem, we can see that it is really quite inferior, mainly because it does not iterate to convergence. The Jacobians are evaluated only once (at the best estimate so far). In truth, the EKF can do better than just one iteration of Gauss-Newton because the EKF does not evaluate all the Jacobians at once, but the lack of iteration is its main downfall. This is obvious from an optimization perspective; we need to iterate to converge. However, the EKF was originally derived from the Bayes filter earlier in this chapter, which used something called the Markov assumption to achieve its recursive form. The problem with the Markov assumption is that once this is built into the estimator, we cannot get rid of it. It is a fundamental constraint that cannot be overcome.

There have been many attempts to patch the EKF, including the Iterated EKF described earlier in this chapter. However, for very nonlinear systems, these may not help much. The problem with the IEKF is that it still clings to the Markov assumption. It is iterating at a single time-step, not over the whole trajectory at once. The difference between Gauss-Newton and the IEKF can be seen plainly in Figure 4.17.

Batch estimation via the Gauss-Newton method has its own problems. In particular, it must be run offline and is not a constant-time algorithm, whereas the EKF is both online and a constant-time method. So-called *sliding-window filters (SWFs)* (Sibley, 2006) seek the best of both worlds by iterating over a window of time-steps and sliding this window along to allow for online/constant-time implementation. SWFs are really still an active area of research, but when viewed from an op-

Figure 4.17
Comparison of the iterative schemes used in various estimation paradigms.

Gauss-Newton iterates over the entire trajectory, but runs offline and not in constant time

$$\mathbf{x}_0 \quad \mathbf{x}_1 \quad \mathbf{x}_2 \quad \mathbf{x}_3 \quad \cdots \quad \mathbf{x}_{k-2} \quad \mathbf{x}_{k-1} \quad \mathbf{x}_k \quad \mathbf{x}_{k+1} \quad \mathbf{x}_{k+2} \quad \cdots \quad \mathbf{x}_K$$



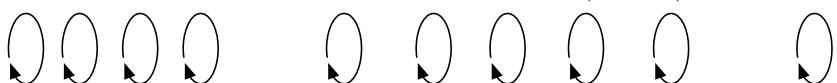
Sliding-window filters iterate over several timesteps at once, run online and in constant time

$$\mathbf{x}_0 \quad \mathbf{x}_1 \quad \mathbf{x}_2 \quad \mathbf{x}_3 \quad \cdots \quad \mathbf{x}_{k-2} \quad \mathbf{x}_{k-1} \quad \mathbf{x}_k \quad \mathbf{x}_{k+1} \quad \mathbf{x}_{k+2} \quad \cdots \quad \mathbf{x}_K$$



IEKF iterates at only one timestep at a time, but runs online and in constant time

$$\mathbf{x}_0 \quad \mathbf{x}_1 \quad \mathbf{x}_2 \quad \mathbf{x}_3 \quad \cdots \quad \mathbf{x}_{k-2} \quad \mathbf{x}_{k-1} \quad \mathbf{x}_k \quad \mathbf{x}_{k+1} \quad \mathbf{x}_{k+2} \quad \cdots \quad \mathbf{x}_K$$



timization perspective, it is hard to imagine that they do not offer a drastic improvement over the EKF and its variants.

4.4 Batch Continuous-Time Estimation

We saw in the previous chapter how to handle continuous-time priors through Gaussian process regression. Our priors were generated by linear stochastic differential equations of the form

$$\dot{\mathbf{x}}(t) = \mathbf{A}(t)\mathbf{x}(t) + \mathbf{v}(t) + \mathbf{L}(t)\mathbf{w}(t), \quad (4.184)$$

with

$$\mathbf{w}(t) \sim \mathcal{GP}(\mathbf{0}, \mathbf{Q} \delta(t - t')), \quad (4.185)$$

and \mathbf{Q} the usual power spectral density matrix.

In this section, we show how we can extend our results to nonlinear, continuous-time motion models of the form

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{v}(t), \mathbf{w}(t), t), \quad (4.186)$$

where $\mathbf{f}(\cdot)$ is a nonlinear function. We will still receive observations at discrete times,

$$\mathbf{y}_k = \mathbf{g}(\mathbf{x}(t_k), \mathbf{n}_k, t), \quad (4.187)$$

where $\mathbf{g}(\cdot)$ is a nonlinear function and

$$\mathbf{n}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k). \quad (4.188)$$

We will begin by linearizing both models and constructing their lifted forms, then carry out GP regression (Bayesian inference). See Anderson et al. (2015) for applications of this section.

4.4.1 Motion Model

We will linearize the motion model about an operating point, $\mathbf{x}_{\text{op}}(t)$, which we note is an entire continuous-time trajectory. We will then construct our motion prior (mean and covariance) in lifted form at the measurement times.

Linearization

Linearizing our motion model about this trajectory, we have

$$\begin{aligned}
\dot{\mathbf{x}}(t) &= \mathbf{f}(\mathbf{x}(t), \mathbf{v}(t), \mathbf{w}(t), t) \\
&\approx \mathbf{f}(\mathbf{x}_{\text{op}}(t), \mathbf{v}(t), \mathbf{0}, t) + \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \Big|_{\mathbf{x}_{\text{op}}(t), \mathbf{v}(t), \mathbf{0}, t} (\mathbf{x}(t) - \mathbf{x}_{\text{op}}(t)) \\
&\quad + \frac{\partial \mathbf{f}}{\partial \mathbf{w}} \Big|_{\mathbf{x}_{\text{op}}(t), \mathbf{v}(t), \mathbf{0}, t} \mathbf{w}(t) \\
&= \underbrace{\mathbf{f}(\mathbf{x}_{\text{op}}(t), \mathbf{v}(t), \mathbf{0}, t) - \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \Big|_{\mathbf{x}_{\text{op}}(t), \mathbf{v}(t), \mathbf{0}, t} \mathbf{x}_{\text{op}}(t)}_{\boldsymbol{\nu}(t)} \\
&\quad + \underbrace{\frac{\partial \mathbf{f}}{\partial \mathbf{x}} \Big|_{\mathbf{x}_{\text{op}}(t), \mathbf{v}(t), \mathbf{0}, t} \mathbf{x}(t)}_{\mathbf{F}(t)} + \underbrace{\frac{\partial \mathbf{f}}{\partial \mathbf{w}} \Big|_{\mathbf{x}_{\text{op}}(t), \mathbf{v}(t), \mathbf{0}, t} \mathbf{w}(t)}, \quad (4.189)
\end{aligned}$$

where $\boldsymbol{\nu}(t)$, $\mathbf{F}(t)$, and $\mathbf{L}(t)$ are now known functions of time (since $\mathbf{x}_{\text{op}}(t)$ is known). Thus, approximately, our process model is of the form

$$\dot{\mathbf{x}}(t) \approx \mathbf{F}(t)\mathbf{x}(t) + \boldsymbol{\nu}(t) + \mathbf{L}(t)\mathbf{w}(t). \quad (4.190)$$

Thus, after linearization, this is in the LTV form we studied in the linear-Gaussian chapter.

Mean and Covariance Functions

Since the SDE for the motion model is approximately in the LTV form studied earlier, we can go ahead and write

$$\begin{aligned}
\mathbf{x}(t) &\sim \mathcal{GP} \left(\underbrace{\Phi(t, t_0)\check{\mathbf{x}}_0 + \int_{t_0}^t \Phi(t, s)\boldsymbol{\nu}(s) ds}_{\check{\mathbf{x}}(t)} \right. \\
&\quad \left. \underbrace{\Phi(t, t_0)\check{\mathbf{P}}_0\Phi(t', t_0)^T + \int_{t_0}^{\min(t, t')} \Phi(t, s)\mathbf{L}(s)\mathbf{Q}\mathbf{L}(s)^T\Phi(t', s)^T ds}_{\check{\mathbf{P}}(t, t')} \right), \quad (4.191)
\end{aligned}$$

where $\Phi(t, s)$ is the transition function associated with $\mathbf{F}(t)$. At the measurement times, $t_0 < t_1 < \dots < t_K$, we can also then write

$$\mathbf{x} \sim \mathcal{N}(\check{\mathbf{x}}, \check{\mathbf{P}}) = \mathcal{N}(\mathbf{F}\boldsymbol{\nu}, \mathbf{F}\mathbf{Q}\mathbf{F}^T), \quad (4.192)$$

for the usual lifted form of the prior where

$$\mathbf{F} = \begin{bmatrix} \mathbf{1} & & & & & \\ \Phi(t_1, t_0) & \mathbf{1} & & & & \\ \Phi(t_2, t_0) & \Phi(t_2, t_1) & \mathbf{1} & & & \\ \vdots & \vdots & \vdots & \ddots & & \\ \Phi(t_{K-1}, t_0) & \Phi(t_{K-1}, t_1) & \Phi(t_{K-1}, t_2) & \cdots & \mathbf{1} & \\ \Phi(t_K, t_0) & \Phi(t_K, t_1) & \Phi(t_K, t_2) & \cdots & \Phi(t_K, t_{K-1}) & \mathbf{1} \end{bmatrix}, \quad (4.193a)$$

$$\boldsymbol{\nu} = \begin{bmatrix} \check{\mathbf{x}}_0 \\ \boldsymbol{\nu}_1 \\ \vdots \\ \boldsymbol{\nu}_K \end{bmatrix}, \quad (4.193b)$$

$$\boldsymbol{\nu}_k = \int_{t_{k-1}}^{t_k} \Phi(t_k, s) \boldsymbol{\nu}(s) ds, \quad k = 1 \dots K, \quad (4.193c)$$

$$\mathbf{Q}' = \text{diag}(\check{\mathbf{P}}_0, \mathbf{Q}'_1, \mathbf{Q}'_2, \dots, \mathbf{Q}'_K), \quad (4.193d)$$

$$\mathbf{Q}'_k = \int_{t_{k-1}}^{t_k} \Phi(t_k, s) \mathbf{L}(s) \mathbf{Q} \mathbf{L}(s)^T \Phi(t_k, s)^T ds, \quad k = 1 \dots K. \quad (4.193e)$$

Unfortunately, we have a bit of a problem. To compute $\check{\mathbf{x}}$ and $\check{\mathbf{P}}$, we require an expression for $\mathbf{x}_{\text{op}}(s)$ for all $s \in [t_0, t_M]$. This is because $\boldsymbol{\nu}(s)$, $\mathbf{F}(s)$ (through $\Phi(t, s)$), and $\mathbf{L}(s)$ appear inside the integrals for $\check{\mathbf{x}}(t)$ and $\check{\mathbf{P}}(t, t')$, and these depend in turn on $\mathbf{x}_{\text{op}}(s)$. If we are performing iterated GP regression, as discussed earlier, we will only have \mathbf{x}_{op} from the previous iteration, which is evaluated only at the measurement times.

Fortunately, the whole point of GP regression is that we can query the state at any time of interest. Moreover, we showed previously that this can be done very efficiently (i.e., $O(1)$ time) for our particular choice of process model:

$$\mathbf{x}_{\text{op}}(s) = \check{\mathbf{x}}(s) + \check{\mathbf{P}}(s) \check{\mathbf{P}}^{-1} (\mathbf{x}_{\text{op}} - \check{\mathbf{x}}). \quad (4.194)$$

Because we are making use of this inside an iterative process, we use the $\check{\mathbf{x}}(s)$, $\check{\mathbf{P}}(s)$, $\check{\mathbf{x}}$, and $\check{\mathbf{P}}$ from the previous iteration to evaluate this expression.

The biggest challenge will be to identify $\Phi(t, s)$, which is problem specific. As we will already be carrying out numerical integration in our scheme, we can compute the transition function numerically as well, via a normalized *fundamental matrix* (of control theory)¹², $\Upsilon(t)$. In other words, we will integrate

$$\dot{\Upsilon}(t) = \mathbf{F}(t) \Upsilon(t), \quad \Upsilon(0) = \mathbf{1}, \quad (4.195)$$

¹² Not to be confused with the fundamental matrix of computer vision.

ensuring to save $\Upsilon(t)$ at all the times of interest in our GP regression. The transition function is then given by

$$\Phi(t, s) = \Upsilon(t) \Upsilon(s)^{-1}. \quad (4.196)$$

For specific systems, analytical expressions for the transition function will be possible.

4.4.2 Observation Model

The linearized observation model is

$$\mathbf{y}_k \approx \mathbf{g}(\mathbf{x}_{\text{op},k}, \mathbf{0}) + \mathbf{G}_k (\mathbf{x}_{k-1} - \mathbf{x}_{\text{op},k-1}) + \mathbf{n}'_k, \quad (4.197)$$

which can be written in lifted form as

$$\mathbf{y} = \mathbf{y}_{\text{op}} + \mathbf{G}(\mathbf{x} - \mathbf{x}_{\text{op}}) + \mathbf{n}', \quad (4.198)$$

where

$$\mathbf{y}_{\text{op}} = \begin{bmatrix} \mathbf{g}(\mathbf{x}_{\text{op},0}, \mathbf{0}) \\ \mathbf{g}(\mathbf{x}_{\text{op},1}, \mathbf{0}) \\ \vdots \\ \mathbf{g}(\mathbf{x}_{\text{op},K}, \mathbf{0}) \end{bmatrix}, \quad (4.199a)$$

$$\mathbf{G} = \text{diag}(\mathbf{G}_0, \mathbf{G}_1, \mathbf{G}_2, \dots, \mathbf{G}_K), \quad (4.199b)$$

$$\mathbf{R} = \text{diag}(\mathbf{R}'_0, \mathbf{R}'_1, \mathbf{R}'_2, \dots, \mathbf{R}'_K), \quad (4.199c)$$

and $\mathbf{n}' \sim \mathcal{N}(\mathbf{0}, \mathbf{R}')$. It is fairly easy to see that

$$E[\mathbf{y}] = \mathbf{y}_{\text{op}} + \mathbf{G}(\check{\mathbf{x}} - \mathbf{x}_{\text{op}}), \quad (4.200a)$$

$$E[(\mathbf{y} - E[\mathbf{y}])(\mathbf{y} - E[\mathbf{y}])^T] = \mathbf{G}\check{\mathbf{P}}\mathbf{G}^T + \mathbf{R}', \quad (4.200b)$$

$$E[(\mathbf{y} - E[\mathbf{y}])(\mathbf{x} - E[\mathbf{x}])^T] = \mathbf{G}\check{\mathbf{P}}. \quad (4.200c)$$

4.4.3 Bayesian Inference

With these quantities in hand, we can write a joint density for the lifted trajectory and measurements as

$$p(\mathbf{x}, \mathbf{y} | \mathbf{v}) = \mathcal{N} \left(\begin{bmatrix} \check{\mathbf{x}} \\ \mathbf{y}_{\text{op}} + \mathbf{G}(\check{\mathbf{x}} - \mathbf{x}_{\text{op}}) \end{bmatrix}, \begin{bmatrix} \check{\mathbf{P}} & \check{\mathbf{P}}\mathbf{G}^T \\ \mathbf{G}\check{\mathbf{P}} & \mathbf{G}\check{\mathbf{P}}\mathbf{G}^T + \mathbf{R}' \end{bmatrix} \right), \quad (4.201)$$

which is quite similar to the expression for the IEKF situation in (4.42), but now for the whole trajectory rather than just one timestep. Using the usual relationship from (2.53b), we can immediately write the Gaussian posterior as

$$p(\mathbf{x} | \mathbf{v}, \mathbf{y}) = \mathcal{N}(\hat{\mathbf{x}}, \hat{\mathbf{P}}), \quad (4.202)$$

where

$$\mathbf{K} = \check{\mathbf{P}}\mathbf{G}^T (\mathbf{G}\check{\mathbf{P}}\mathbf{G}^T + \mathbf{R}')^{-1}, \quad (4.203a)$$

$$\hat{\mathbf{P}} = (\mathbf{I} - \mathbf{K}\mathbf{G})\check{\mathbf{P}}, \quad (4.203b)$$

$$\hat{\mathbf{x}} = \check{\mathbf{x}} + \mathbf{K}(\mathbf{y} - \mathbf{y}_{op} - \mathbf{G}(\check{\mathbf{x}} - \mathbf{x}_{op})). \quad (4.203c)$$

Using the SMW identity from (2.75), we can rearrange the equation for the posterior mean to be

$$(\check{\mathbf{P}}^{-1} + \mathbf{G}^T \mathbf{R}'^{-1} \mathbf{G}) \delta \mathbf{x}^* = \check{\mathbf{P}}^{-1} (\check{\mathbf{x}} - \mathbf{x}_{op}) + \mathbf{G}^T \mathbf{R}'^{-1} (\mathbf{y} - \mathbf{y}_{op}), \quad (4.204)$$

where $\delta \mathbf{x}^* = \hat{\mathbf{x}} - \mathbf{x}_{op}$. Inserting the details of the prior, this becomes

$$\underbrace{(\mathbf{F}^{-T} \mathbf{Q}'^{-1} \mathbf{F}^{-1} + \mathbf{G}^T \mathbf{R}'^{-1} \mathbf{G})}_{\text{block-tridiagonal}} \delta \mathbf{x}^* = \mathbf{F}^{-T} \mathbf{Q}'^{-1} (\boldsymbol{\nu} - \mathbf{F}^{-1} \mathbf{x}_{op}) + \mathbf{G}^T \mathbf{R}'^{-1} (\mathbf{y} - \mathbf{y}_{op}). \quad (4.205)$$

This result is identical in form to the nonlinear, discrete-time batch solution discussed earlier in this chapter. The only difference is that we started with a continuous-time motion model and integrated it directly to evaluate the prior at the measurement times.

4.4.4 Algorithm Summary

We summarize the steps needed to carry out GP regression with a nonlinear motion model and/or measurement model:

1. Start with an initial guess for the posterior mean over the whole trajectory, $\mathbf{x}_{op}(t)$. We will only need this over the whole trajectory to initialize the process. We will only be updating our estimate at the measurement times, \mathbf{x}_{op} , then using the GP interpolation to fill in the other times.
2. Calculate the $\boldsymbol{\nu}$, \mathbf{F}^{-1} , and \mathbf{Q}'^{-1} for the new iteration. This will likely be done numerically and will involve determining $\boldsymbol{\nu}(s)$, $\mathbf{F}(s)$ (through $\Phi(t, s)$), and $\mathbf{L}(s)$, which in turn will require $\mathbf{x}_{op}(s)$ and hence $\check{\mathbf{x}}(s)$, $\mathbf{P}(s)$, $\check{\mathbf{x}}$, $\check{\mathbf{P}}$ from the previous iteration to do the interpolation inside the required integrals.
3. Calculate the \mathbf{y}_{op} , \mathbf{G} , \mathbf{R}'^{-1} for the new iteration.
4. Solve for $\delta \mathbf{x}^*$ in the following equation:

$$\underbrace{(\mathbf{F}^{-T} \mathbf{Q}'^{-1} \mathbf{F}^{-1} + \mathbf{G}^T \mathbf{R}'^{-1} \mathbf{G})}_{\text{block-tridiagonal}} \delta \mathbf{x}^* = \mathbf{F}^{-T} \mathbf{Q}'^{-1} (\boldsymbol{\nu} - \mathbf{F}^{-1} \mathbf{x}_{op}) + \mathbf{G}^T \mathbf{R}'^{-1} (\mathbf{y} - \mathbf{y}_{op}). \quad (4.206)$$

In practice, we will prefer to build only the non-zero blocks in the products of matrices that appear in this equation.

5. Update the guess at the measurement times using

$$\mathbf{x}_{\text{op}} \leftarrow \mathbf{x}_{\text{op}} + \delta \mathbf{x}^*, \quad (4.207)$$

and check for convergence. If not converged, return to Step 2. If converged, output $\hat{\mathbf{x}} = \mathbf{x}_{\text{op}}$.

6. If desired, compute the covariance at the measurement times, $\hat{\mathbf{P}}$.
7. Use the GP interpolation equation¹³ also to compute the estimate at other times of interest, $\hat{\mathbf{x}}_\tau$, $\hat{\mathbf{P}}_{\tau\tau}$.

The most expensive step in this whole process is building $\boldsymbol{\nu}$, \mathbf{F}^{-1} , and \mathbf{Q}'^{-1} . However, the cost (at each iteration) will still be linear in the length of the trajectory and therefore should be manageable.

4.5 Summary

The main take-away points from this chapter are as follows:

1. Unlike the linear-Gaussian case, the Bayesian posterior is not, in general, a Gaussian PDF when the motion and observation models are nonlinear and/or the measurement and process noises are non-Gaussian.
2. To carry out nonlinear estimation, some form of approximation is required. The different techniques vary in their choices of (i) how to approximate the posterior (Gaussian, mixture of Gaussians, set of samples), and (ii) how to approximately carry out inference (linearization, Monte Carlo, sigma point transformation) or MAP estimation.
3. There are a variety of methods, both batch and recursive, that approximate the posterior as a Gaussian. Some of these methods, particularly the ones that iterate the solution (i.e., batch MAP, IEKF) converge to a ‘mean’ that is actually at the maximum of the Bayesian posterior (which is not the same as the true mean of the Bayesian posterior). This can be a point of confusion when comparing different methods since, depending on the approximations made, we may be asking the methods to find different answers.
4. Batch methods are able to iterate over the whole trajectory, whereas recursive methods can only iterate at one time-step at a time, meaning they will converge to different answers on most problems.

The next chapter will look briefly at how to handle estimator bias, measurement outliers, and data correspondences.

¹³ We did not work this out for the nonlinear case, but it should follow from the GP section in the linear-Gaussian chapter.

4.6 Exercises

4.6.1 Consider the discrete-time system,

$$\begin{aligned} \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} &= \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \theta_{k-1} \end{bmatrix} + T \begin{bmatrix} \cos \theta_{k-1} & 0 \\ \sin \theta_{k-1} & 0 \\ 0 & 1 \end{bmatrix} \left(\begin{bmatrix} v_k \\ \omega_k \end{bmatrix} + \mathbf{w}_k \right), \\ \mathbf{w}_k &\sim \mathcal{N}(\mathbf{0}, \mathbf{Q}), \\ \begin{bmatrix} r_k \\ \phi_k \end{bmatrix} &= \begin{bmatrix} \sqrt{x_k^2 + y_k^2} \\ \text{atan2}(-y_k, -x_k) - \theta_k \end{bmatrix} + \mathbf{n}_k, \quad \mathbf{n}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}), \end{aligned}$$

which could represent a mobile robot moving around on the xy -plane and measuring the range and bearing to the origin. Set up the EKF equation to estimate the pose of the mobile robot. In particular, work out expressions for the Jacobians, \mathbf{F}_{k-1} and \mathbf{G}_k , and modified covariances, \mathbf{Q}'_k and \mathbf{R}'_k .

- 4.6.2 Consider transforming the prior Gaussian density, $\mathcal{N}(\mu_x, \sigma_x^2)$, through the nonlinearity, $f(x) = x^3$. Use the Monte Carlo, linearization, and sigmapoint methods to determine the transformed mean and covariance and comment on the results. Hint: use Isserlis' theorem to compute the higher-order Gaussian moments.
- 4.6.3 Consider transforming the prior Gaussian density, $\mathcal{N}(\mu_x, \sigma_x^2)$, through the nonlinearity, $f(x) = x^4$. Use the Monte Carlo, linearization, and sigmapoint methods to determine the transformed mean (and optionally covariance) and comment on the results. Hint: use Isserlis' theorem to compute the higher-order Gaussian moments.
- 4.6.4 From the section on the sigmapoint Kalman filter, we learned that the measurement covariance could be written as

$$\Sigma_{yy,k} = \sum_{j=0}^{2N} \beta_j (\check{\mathbf{y}}_{k,j} - \boldsymbol{\mu}_{y,k}) (\check{\mathbf{y}}_{k,j} - \boldsymbol{\mu}_{y,k})^T + \mathbf{R}_k,$$

when the measurement model has linear dependence on the measurement noise. Verify that this can also be written as

$$\Sigma_{yy,k} = \mathbf{Z}_k \mathbf{Z}_k^T + \mathbf{R}_k,$$

where

$$\text{col}_j \mathbf{Z}_k = \sqrt{\beta_j} (\check{\mathbf{y}}_{k,j} - \boldsymbol{\mu}_{y,k}).$$

- 4.6.5 Show that the below two identities used in the section on ML bias estimation are true:

$$\begin{aligned} \mathbf{A}(\mathbf{x}) \mathbf{R} \mathbf{A}(\mathbf{x})^T &\equiv \mathbf{W}(\mathbf{x})^{-1}, \\ \mathbf{A}(\mathbf{x}) \mathbf{R} \mathbf{P}_k^T &\equiv \mathbf{W}(\mathbf{x})^{-1} \mathbf{G}_k(\mathbf{x})^T. \end{aligned}$$

5

Biases, Correspondences, and Outliers

In the last chapter, we learned that our estimation machinery can be biased, particularly when our motion/observation models are nonlinear. In our simple stereo camera example, we saw that MAP estimation is biased with respect to the mean of the full posterior. We also saw that the batch ML method is biased with respect to the groundtruth and derived an expression to try to quantify that bias. Unfortunately, these are not the only sources of bias.

In many of our estimation techniques, we make the assumption that the noise corrupting the inputs or the measurements is zero-mean Gaussian. In reality, our inputs and/or measurements may also be corrupted with unknown biases. If we do not account for these, our estimate will also be biased. The classic example of this is the typical accelerometer, which can have temperature-dependent biases that change over time.

Another huge issue in many estimation problems is determining correspondences between measurements and a model. For example, if we are measuring the range to a landmark, we might assume we know which landmark is being measured. This is a very big assumption. Another good example is a star tracker, which detects points of lights; how do we know which point of light corresponds to which star in our star chart? The pairing of a measurement with a part of a model/map is termed *determining correspondences* or *data association*.

Finally, despite our best efforts to negate the effects of biases and find proper correspondences, something deleterious can always happen to our measurements so that we are stuck with a datum that is highly improbable according to our noise model; we call this an *outlier* measurement. If we do not properly detect and remove outliers, many of our estimation techniques will fail, often catastrophically.

This chapter will investigate how to deal with inputs/measurements that are not well behaved. It will present some of the classic tactics for handling these types of biases, determining correspondences, and detecting/rejecting outliers. A handful of examples will be provided as illustrations.

5.1 Handling Input/Measurement Biases

In this section, we will investigate the impact of a bias on both the inputs and measurements. We will see that the case of the input bias is less difficult to deal with than the measurement bias, but both can be handled. We will use linear, time-invariant motion and observation models with non-zero-mean Gaussian noise for the purpose of our discussion, but many of the concepts to be discussed can also be extended to nonlinear systems.

5.1.1 Bias Effects on the Kalman Filter

As an example of the effect of a input/measurement bias, we return to the error dynamics discussed in Section 3.3.6 and see what happens to the Kalman filter (if we do not explicitly account for bias) when we introduce non-zero-mean noise. In particular, we will now assume that

$$\mathbf{x}_k = \mathbf{A}\mathbf{x}_{k-1} + \mathbf{B}(\mathbf{u}_k + \bar{\mathbf{u}}) + \mathbf{w}_k, \quad (5.1a)$$

$$\mathbf{y}_k = \mathbf{C}\mathbf{x}_k + \bar{\mathbf{y}} + \mathbf{n}_k, \quad (5.1b)$$

where $\bar{\mathbf{u}}$ is an input bias and $\bar{\mathbf{y}}$ a measurement bias. We will continue to assume that all measurements are corrupted with zero-mean Gaussian noise,

$$\mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}), \quad \mathbf{n}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}), \quad (5.2)$$

that is statistically independent, i.e.,

$$E[\mathbf{w}_k \mathbf{w}_\ell^T] = \mathbf{0}, \quad E[\mathbf{n}_k \mathbf{n}_\ell^T] = \mathbf{0}, \quad E[\mathbf{w}_k \mathbf{n}_\ell^T] = \mathbf{0}, \quad E[\mathbf{w}_k \mathbf{n}_k^T] = \mathbf{0}, \quad (5.3)$$

for all $k \neq \ell$, but this could be another source of filter inconsistency. We earlier defined the estimation errors,

$$\check{\mathbf{e}}_k = \check{\mathbf{x}}_k - \mathbf{x}_k, \quad (5.4a)$$

$$\hat{\mathbf{e}}_k = \hat{\mathbf{x}}_k - \mathbf{x}_k, \quad (5.4b)$$

and constructed the ‘error dynamics’, which in this case are

$$\check{\mathbf{e}}_k = \mathbf{A}\hat{\mathbf{e}}_{k-1} - (\mathbf{B}\bar{\mathbf{u}} + \mathbf{w}_k), \quad (5.5a)$$

$$\hat{\mathbf{e}}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{C}) \check{\mathbf{e}}_k + \mathbf{K}_k (\bar{\mathbf{y}} + \mathbf{n}_k). \quad (5.5b)$$

where $\hat{\mathbf{e}}_0 = \hat{\mathbf{x}}_0 - \mathbf{x}_0$. Furthermore, as discussed earlier, for our estimator to be *unbiased* and *consistent* we would like to have for all $k = 1 \dots K$ that

$$E[\hat{\mathbf{e}}_k] = \mathbf{0}, \quad E[\check{\mathbf{e}}_k] = \mathbf{0}, \quad E[\hat{\mathbf{e}}_k \hat{\mathbf{e}}_k^T] = \hat{\mathbf{P}}_k, \quad E[\check{\mathbf{e}}_k \check{\mathbf{e}}_k^T] = \check{\mathbf{P}}_k, \quad (5.6)$$

which we showed was true in the case that $\bar{\mathbf{u}} = \bar{\mathbf{y}} = \mathbf{0}$. Let us see what happens when this zero-bias condition does not necessarily hold. We

will still assume that

$$E[\hat{\mathbf{e}}_0] = \mathbf{0}, \quad E[\hat{\mathbf{e}}_0 \hat{\mathbf{e}}_0^T] = \hat{\mathbf{P}}_0, \quad (5.7)$$

although this initial condition is another place a bias could be introduced. At $k = 1$ we have

$$E[\check{\mathbf{e}}_1] = \underbrace{\mathbf{A} E[\hat{\mathbf{e}}_0]}_{\mathbf{0}} - \left(\mathbf{B} \bar{\mathbf{u}} + \underbrace{E[\mathbf{w}_1]}_{\mathbf{0}} \right) = -\mathbf{B} \bar{\mathbf{u}}, \quad (5.8a)$$

$$\begin{aligned} E[\hat{\mathbf{e}}_1] &= (\mathbf{1} - \mathbf{K}_1 \mathbf{C}) \underbrace{E[\check{\mathbf{e}}_1]}_{-\mathbf{B} \bar{\mathbf{u}}} + \mathbf{K}_1 \left(\bar{\mathbf{y}} + \underbrace{E[\mathbf{n}_1]}_{\mathbf{0}} \right) \\ &= -(\mathbf{1} - \mathbf{K}_1 \mathbf{C}) \mathbf{B} \bar{\mathbf{u}} + \mathbf{K}_1 \bar{\mathbf{y}}, \end{aligned} \quad (5.8b)$$

which are already biased in the case that $\bar{\mathbf{u}} \neq \mathbf{0}$ and/or $\bar{\mathbf{y}} \neq \mathbf{0}$. For the covariance of the ‘predicted error’ we have

$$\begin{aligned} E[\check{\mathbf{e}}_1 \check{\mathbf{e}}_1^T] &= E[(\mathbf{A} \hat{\mathbf{e}}_0 - (\mathbf{B} \bar{\mathbf{u}} + \mathbf{w}_1)) (\mathbf{A} \hat{\mathbf{e}}_0 - (\mathbf{B} \bar{\mathbf{u}} + \mathbf{w}_1))^T] \\ &= \underbrace{E[(\mathbf{A} \hat{\mathbf{e}}_0 - \mathbf{w}_1)(\mathbf{A} \hat{\mathbf{e}}_0 - \mathbf{w}_1)^T]}_{\check{\mathbf{P}}_1} + (-\mathbf{B} \bar{\mathbf{u}}) \underbrace{E[(\mathbf{A} \hat{\mathbf{e}}_0 - \mathbf{w}_1)^T]}_{\mathbf{0}} \\ &\quad + \underbrace{E[(\mathbf{A} \hat{\mathbf{e}}_0 - \mathbf{w}_1)]}_{\mathbf{0}} (-\mathbf{B} \bar{\mathbf{u}})^T + (-\mathbf{B} \bar{\mathbf{u}})(-\mathbf{B} \bar{\mathbf{u}})^T \\ &= \check{\mathbf{P}}_1 + (-\mathbf{B} \bar{\mathbf{u}})(-\mathbf{B} \bar{\mathbf{u}})^T. \end{aligned} \quad (5.9)$$

Rearranging, we see that

$$\check{\mathbf{P}}_1 = E[\check{\mathbf{e}}_1 \check{\mathbf{e}}_1^T] - \underbrace{E[\check{\mathbf{e}}_1] E[\check{\mathbf{e}}_1]^T}_{\text{bias effect}}, \quad (5.10)$$

and therefore the KF will ‘underestimate’ the true uncertainty in the error and become inconsistent. For the covariance of the ‘corrected

error' we have

$$\begin{aligned}
E[\hat{\mathbf{e}}_1 \hat{\mathbf{e}}_1^T] &= E \left[((\mathbf{1} - \mathbf{K}_1 \mathbf{C}) \check{\mathbf{e}}_1 + \mathbf{K}_1 (\bar{\mathbf{y}} + \mathbf{n}_1)) \right. \\
&\quad \times \left. ((\mathbf{1} - \mathbf{K}_1 \mathbf{C}) \check{\mathbf{e}}_1 + \mathbf{K}_1 (\bar{\mathbf{y}} + \mathbf{n}_1))^T \right] \\
&= \underbrace{E \left[((\mathbf{1} - \mathbf{K}_1 \mathbf{C}) \check{\mathbf{e}}_1 + \mathbf{K}_1 \mathbf{n}_1) ((\mathbf{1} - \mathbf{K}_1 \mathbf{C}) \check{\mathbf{e}}_1 + \mathbf{K}_1 \mathbf{n}_1)^T \right]}_{\hat{\mathbf{P}}_1 + (\mathbf{1} - \mathbf{K}_1 \mathbf{C}) \mathbf{B} \bar{\mathbf{u}} \bar{\mathbf{u}}^T \mathbf{B}^T (\mathbf{1} - \mathbf{K}_1 \mathbf{C})^T} \\
&\quad + (\mathbf{K}_1 \bar{\mathbf{y}}) \underbrace{E \left[((\mathbf{1} - \mathbf{K}_1 \mathbf{C}) \check{\mathbf{e}}_1 + \mathbf{K}_1 \mathbf{n}_1)^T \right]}_{(-(\mathbf{1} - \mathbf{K}_1 \mathbf{C}) \mathbf{B} \bar{\mathbf{u}})^T} \\
&\quad + \underbrace{E \left[((\mathbf{1} - \mathbf{K}_1 \mathbf{C}) \check{\mathbf{e}}_1 + \mathbf{K}_1 \mathbf{n}_1) \right]}_{-(\mathbf{1} - \mathbf{K}_1 \mathbf{C}) \mathbf{B} \bar{\mathbf{u}}} (\mathbf{K}_1 \bar{\mathbf{y}})^T + (\mathbf{K}_1 \bar{\mathbf{y}}) (\mathbf{K}_1 \bar{\mathbf{y}})^T \\
&= \hat{\mathbf{P}}_1 + (-(\mathbf{1} - \mathbf{K}_1 \mathbf{C}) \mathbf{B} \bar{\mathbf{u}} + \mathbf{K}_1 \bar{\mathbf{y}}) \\
&\quad \times (-(\mathbf{1} - \mathbf{K}_1 \mathbf{C}) \mathbf{B} \bar{\mathbf{u}} + \mathbf{K}_1 \bar{\mathbf{y}})^T, \tag{5.11}
\end{aligned}$$

and so

$$\hat{\mathbf{P}}_1 = E[\hat{\mathbf{e}}_1 \hat{\mathbf{e}}_1^T] - \underbrace{E[\hat{\mathbf{e}}_1] E[\hat{\mathbf{e}}_1]^T}_{\text{bias effect}}, \tag{5.12}$$

where we can see again that the KF's estimate of the covariance is overconfident and thus inconsistent. It is interesting to note that the KF will become overconfident, regardless of the sign of the bias. Moreover, it is not hard to see that as k gets bigger, the effects of the biases grow without bound. It is tempting to modify the KF to be

$$\text{predictor: } \check{\mathbf{x}}_k = \mathbf{A} \hat{\mathbf{P}}_{k-1} \mathbf{A}^T + \mathbf{Q}, \tag{5.13a}$$

$$\check{\mathbf{x}}_k = \mathbf{A} \hat{\mathbf{x}}_{k-1} + \mathbf{B} \mathbf{u}_k + \underbrace{\mathbf{B} \bar{\mathbf{u}}}_{\text{bias}}, \tag{5.13b}$$

$$\text{Kalman gain: } \mathbf{K}_k = \check{\mathbf{P}}_k \mathbf{C}^T (\mathbf{C} \check{\mathbf{P}}_k \mathbf{C}^T + \mathbf{R})^{-1}, \tag{5.13c}$$

$$\text{corrector: } \hat{\mathbf{P}}_k = (\mathbf{1} - \mathbf{K}_k \mathbf{C}) \check{\mathbf{P}}_k, \tag{5.13d}$$

$$\hat{\mathbf{x}}_k = \check{\mathbf{x}}_k + \mathbf{K}_k \left(\mathbf{y}_k - \mathbf{C} \check{\mathbf{x}}_k - \underbrace{\bar{\mathbf{y}}}_{\text{bias}} \right), \tag{5.13e}$$

whereupon we recover an unbiased and consistent estimate. The problem is that we must know the value of the bias exactly for this to be a viable means of counteracting its effects. In most cases we do not know the exact value of the bias (it may even change with time). Given that we already have an estimation problem, it is logical to attempt to include estimation of the bias into our problem. The next few sections will investigate this possibility for both inputs and measurements.

5.1.2 Unknown Input Bias

Continuing from the previous section, suppose we had $\bar{\mathbf{y}} = \mathbf{0}$ but not necessarily $\bar{\mathbf{u}} \neq \mathbf{0}$. Rather than estimating just the state of the system, \mathbf{x}_k , we augment the state to be

$$\mathbf{x}'_k = \begin{bmatrix} \mathbf{x}_k \\ \bar{\mathbf{u}}_k \end{bmatrix}, \quad (5.14)$$

where we have made the bias now a function of time as we want it to be part of our state. As the bias is now a function of time, we need to define a motion model for it. A typical one is to assume that

$$\bar{\mathbf{u}}_k = \bar{\mathbf{u}}_{k-1} + \mathbf{s}_k, \quad (5.15)$$

where $\mathbf{s}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{W})$; this corresponds to Brownian motion (a.k.a., random walk) of the bias. In some sense, we are simply pushing the problem back through an integrator as we now have zero-mean Gaussian noise influencing the motion of the interoceptive bias. In practice, this type of trick can be effective. Other motion models for the bias could also be assumed, but often we do not have a lot of information as to their temporal behaviour. Under this bias motion model, we have for the motion model for our augmented state that

$$\mathbf{x}'_k = \underbrace{\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{0} & \mathbf{1} \end{bmatrix}}_{\mathbf{A}'} \mathbf{x}'_{k-1} + \underbrace{\begin{bmatrix} \mathbf{B} \\ \mathbf{0} \end{bmatrix}}_{\mathbf{B}'} \mathbf{u}_k + \underbrace{\begin{bmatrix} \mathbf{w}_k \\ \mathbf{s}_k \end{bmatrix}}_{\mathbf{w}'_k}, \quad (5.16)$$

where we have defined several new symbols for convenience. We note that

$$\mathbf{w}'_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}'), \quad \mathbf{Q}' = \begin{bmatrix} \mathbf{Q} & \mathbf{0} \\ \mathbf{0} & \mathbf{W} \end{bmatrix}, \quad (5.17)$$

so we are back to an unbiased system. The observation model is simply

$$\mathbf{y}_k = \underbrace{\begin{bmatrix} \mathbf{C} & \mathbf{0} \end{bmatrix}}_{\mathbf{C}'} \mathbf{x}'_k + \mathbf{n}_k, \quad (5.18)$$

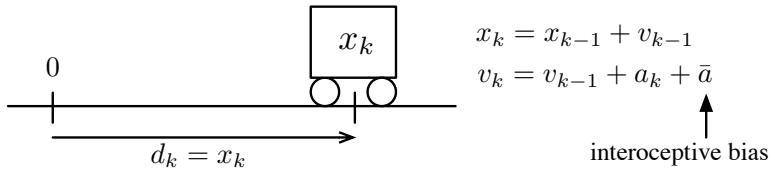
in terms of the augmented state.

A critical question to ask is whether or not this augmented-state filter will converge to the correct answer. Will the above trick really work? The conditions we saw for existence and uniqueness of the linear-Gaussian batch estimation earlier (with no prior on the initial condition) were

$$\mathbf{Q} > 0, \quad \mathbf{R} > 0, \quad \text{rank } \mathcal{O} = N. \quad (5.19)$$

Let us assume these conditions do indeed hold for the system in the

Figure 5.1 Input bias on acceleration. In this case we can successfully estimate the bias as part of our state estimation problem.



case that the bias is zero, i.e., $\bar{\mathbf{u}} = \mathbf{0}$. Defining

$$\mathcal{O}' = \begin{bmatrix} \mathbf{C}' \\ \mathbf{C}'\mathbf{A}' \\ \vdots \\ \mathbf{C}'\mathbf{A}'^{(N+U-1)} \end{bmatrix}, \quad (5.20)$$

we are required to show that

$$\underbrace{\mathbf{Q}' > 0, \quad \mathbf{R} > 0}_{\text{true by definitions}}, \quad \text{rank } \mathcal{O}' = N + U, \quad (5.21)$$

for existence and uniqueness of the solution to the batch estimation problem for the augmented-state system. The first two conditions are true by the definitions of these covariance matrices. For the last condition, the rank needs to be $N + U$ since the augmented state now includes the bias, where $U = \dim \bar{\mathbf{u}}_k$. In general this condition does not hold. The next two examples will illustrate this.

Example 5.1 Take the system matrices to be

$$\mathbf{A} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \mathbf{C} = [1 \ 0], \quad (5.22)$$

such that $N = 2$ and $U = 1$. This example roughly corresponds to a simple one-dimensional unit-mass cart, whose state is its position and velocity. The input is the acceleration and the measurement is the distance back to the origin. The bias is on the input. See Figure 5.1 for an illustration. We have

$$\mathcal{O} = \begin{bmatrix} \mathbf{C} \\ \mathbf{C}\mathbf{A} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \Rightarrow \text{rank } \mathcal{O} = 2 = N, \quad (5.23)$$

so the unbiased system is observable¹. For the augmented-state system we have

$$\begin{aligned} \mathcal{O}' &= \begin{bmatrix} \mathbf{C}' \\ \mathbf{C}'\mathbf{A}' \\ \mathbf{C}'\mathbf{A}'^2 \end{bmatrix} = \begin{bmatrix} \mathbf{C} & \mathbf{0} \\ \mathbf{C}\mathbf{A} & \mathbf{C}\mathbf{B} \\ \mathbf{C}\mathbf{A}^2 & \mathbf{CAB} + \mathbf{CB} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 2 & 1 \end{bmatrix} \\ &\Rightarrow \text{rank } \mathcal{O}' = 3 = N + U, \quad (5.24) \end{aligned}$$

¹ It is also *controllable*.

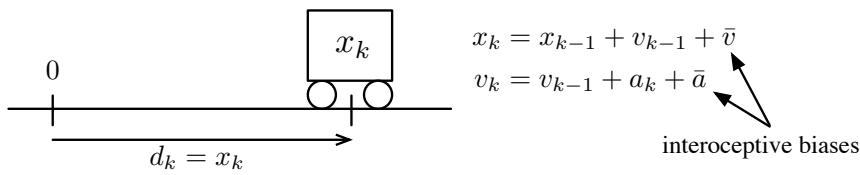


Figure 5.2 Input biases on both speed and acceleration. In this case we cannot estimate the bias as the system is unobservable.

so it, too, is observable. Note that taking $\mathbf{B} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ is observable, too².

Example 5.2 Take the system matrices to be

$$\mathbf{A} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \mathbf{C} = [1 \ 0], \quad (5.25)$$

such that $N = 2$ and $U = 2$. This is a strange system wherein the command to the system is a function of both speed and acceleration, and we have biases on both of these quantities. See Figure 5.2 for an illustration. We still have that the unbiased system is observable since \mathbf{A} and \mathbf{C} are unchanged. For the augmented-state system we have

$$\begin{aligned} \mathcal{O}' = \begin{bmatrix} \mathbf{C}' \\ \mathbf{C}'\mathbf{A}' \\ \mathbf{C}'\mathbf{A}'^2 \\ \mathbf{C}'\mathbf{A}'^3 \end{bmatrix} &= \begin{bmatrix} \mathbf{C} & \mathbf{0} \\ \mathbf{C}\mathbf{A} & \mathbf{C}\mathbf{B} \\ \mathbf{C}\mathbf{A}^2 & \mathbf{C}(\mathbf{A}+\mathbf{1})\mathbf{B} \\ \mathbf{C}\mathbf{A}^3 & \mathbf{C}(\mathbf{A}^2+\mathbf{A}+\mathbf{1})\mathbf{B} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 2 & 2 & 1 \\ 1 & 3 & 3 & 3 \end{bmatrix} \\ &\Rightarrow \text{rank } \mathcal{O}' = 3 < 4 = N + U, \quad (5.26) \end{aligned}$$

so it is not observable (since columns 2 and 3 are the same).

5.1.3 Unknown Measurement Bias

Suppose now we have $\bar{\mathbf{u}} = \mathbf{0}$ but not necessarily $\bar{\mathbf{y}} \neq \mathbf{0}$. The augmented state is

$$\mathbf{x}'_k = \begin{bmatrix} \mathbf{x}_k \\ \bar{\mathbf{y}}_k \end{bmatrix}, \quad (5.27)$$

where we have again made the bias a function of time. We again assume a random-walk motion model

$$\bar{\mathbf{y}}_k = \bar{\mathbf{y}}_{k-1} + \mathbf{s}_k, \quad (5.28)$$

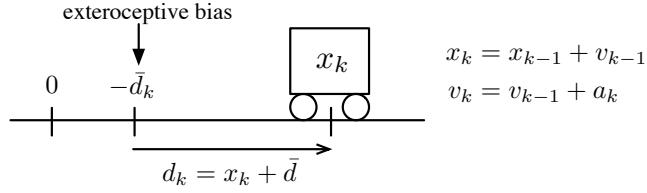
where $\mathbf{s}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{W})$. Under this bias motion model, we have for the motion model for our augmented state that

$$\mathbf{x}'_k = \underbrace{\begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix}}_{\mathbf{A}'} \mathbf{x}'_{k-1} + \underbrace{\begin{bmatrix} \mathbf{B} \\ \mathbf{0} \end{bmatrix}}_{\mathbf{B}'} \mathbf{u}_k + \underbrace{\begin{bmatrix} \mathbf{w}_k \\ \mathbf{s}_k \end{bmatrix}}_{\mathbf{w}'_k}, \quad (5.29)$$

² But now the unbiased system is not *controllable*.

Figure 5.3

Measurement bias on position. In this case we cannot estimate the bias as the system is unobservable.



where we have defined several new symbols for convenience. We note that

$$\mathbf{w}'_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}') , \quad \mathbf{Q}' = \begin{bmatrix} \mathbf{Q} & \mathbf{0} \\ \mathbf{0} & \mathbf{W} \end{bmatrix}. \quad (5.30)$$

The observation model is

$$\mathbf{y}_k = \underbrace{[\mathbf{C} \quad \mathbf{1}]}_{\mathbf{C}'} \mathbf{x}'_k + \mathbf{n}_k, \quad (5.31)$$

in terms of the augmented state. We again examine the observability of the system in the context of an example.

Example 5.3 Take the system matrices to be

$$\mathbf{A} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \mathbf{C} = [1 \quad 0], \quad (5.32)$$

such that $N = 2$ and $U = 1$. This corresponds to our cart measuring its distance from a landmark (whose position it does not know – see Figure 5.3). In the context of mobile robotics this is a very simple example of *simultaneous localization and mapping (SLAM)*, a popular estimation research area. The ‘localization’ is the cart state and the ‘map’ is the landmark position (here the negative of the bias).

We have

$$\mathcal{O} = \begin{bmatrix} \mathbf{C} \\ \mathbf{CA} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \Rightarrow \text{rank } \mathcal{O} = 2 = N, \quad (5.33)$$

so the unbiased system is observable. For the augmented-state system we have

$$\begin{aligned} \mathcal{O}' &= \begin{bmatrix} \mathbf{C}' \\ \mathbf{C}'\mathbf{A}' \\ \mathbf{C}'\mathbf{A}'^2 \end{bmatrix} = \begin{bmatrix} \mathbf{C} & \mathbf{1} \\ \mathbf{CA} & \mathbf{1} \\ \mathbf{CA}^2 & \mathbf{1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 2 & 1 \end{bmatrix} \\ &\Rightarrow \text{rank } \mathcal{O}' = 2 < 3 = N + U, \quad (5.34) \end{aligned}$$

so it is not observable (since columns 1 and 3 are the same). Since we are rank-deficient by 1, this means that $\dim(\text{null } \mathcal{O}') = 1$; the nullspace of the observability matrix corresponds to those vectors that

produce outputs of zero. Here we see that

$$\text{null } \mathcal{O}' = \text{span} \left\{ \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \right\}, \quad (5.35)$$

which means that we can shift the cart and landmark together (left or right) and the measurement will not change. Does this mean our estimator will fail? Not if we are careful to interpret the solutions properly; we do so for both batch LG estimation and the KF:

- (i) In the batch LG estimator, the left-hand side cannot be inverted, but recalling basic linear algebra, every system of the form $\mathbf{Ax} = \mathbf{b}$ can have zero, one, or infinitely many solutions. In this case we have infinitely many solutions rather than a single unique solution.
- (ii) In the KF, we need to start with an initial guess for the state. The final answer we get will depend on the initial conditions selected. In other words, the value of the bias will remain at its initial guess.

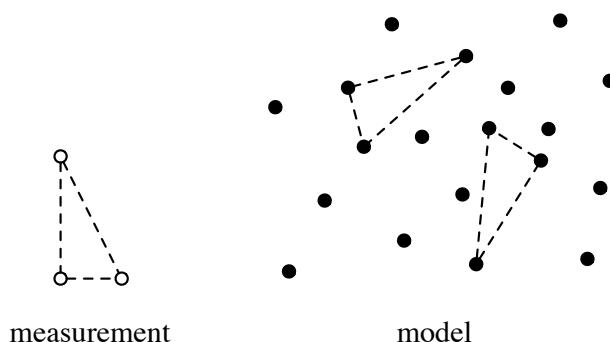
In both cases, we have a way forward.

5.2 Data Association

As discussed above, the *data association* problem has to do with figuring out which measurements correspond to which parts of a model. Virtually all real estimation techniques, particularly for robotics, employ some form of model or map to determine a vehicle's state, and in particular its position/orientation. Some common examples of these models/maps are as follows:

- (i) Positioning using GPS satellites. Here the positions of the GPS satellites are assumed to be known (as a function of time) in a reference frame attached to the Earth (e.g., using their orbital elements). A GPS receiver on the ground measures range to the satellites (e.g., using time of flight based on a timing message sent by the satellite) and then trilaterates for position. In this case, it is easy to know which range measurement is associated with which satellite because the whole system is engineered and therefore unique codes are embedded in the timing messages to indicate which satellite has sent which message.
- (ii) Attitude determination using celestial observation. A map (or chart) of all the brightest stars in the sky is used by a star sensor to determine which direction the sensor is pointing. Here the natural world is being used as a map (surveyed in advance) and thus data association, or knowing which star you are looking

Figure 5.4 A measurement and point-cloud model with two possible data associations shown.



at, is much more difficult than in the GPS case. Because the star chart can be generated in advance, this system becomes practical.

There are essentially two main approaches to data association: external and internal.

5.2.1 External Data Association

In external data association, specialized knowledge of the model/measurements is used for association. This knowledge is ‘external’ to the estimation problem. This is sometimes called ‘known data association’ because from the perspective of the estimation problem, the job has been done.

For example, a bunch of targets could be painted with unique colours; a stereo camera could be used to observe the targets and the colour information used to do data association. The colour information would not be used in the estimation problem. Other examples of external data association include visual bar codes and unique transmission frequencies/codes (e.g., GPS satellites).

External data association can work well if the model can be modified in advance to be cooperative; this makes the estimation problem a lot easier. However, cutting-edge computer vision techniques can be used as external data association on unprepared models, too, although the results are more prone to misassociations.

5.2.2 Internal Data Association

In internal data association, only the measurements/model are used to do data association. This is sometimes called ‘unknown data association’. Typically, association is based on the likelihood of a given measurement, given the model. In the simplest version, the most likely association is accepted and the other possibilities are ignored. More

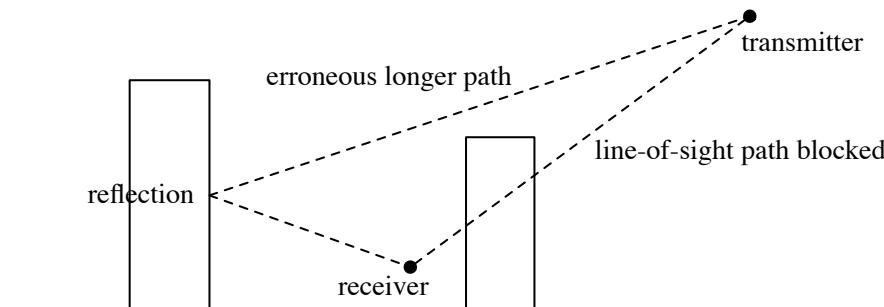


Figure 5.5 A pathological configuration of buildings can trick a GPS system into using an incorrect range measurement.

sophisticated techniques allow multiple data association hypotheses to be carried forward into the estimation problem.

In the case of certain types of models, such as three-dimensional landmarks or star charts, ‘constellations’ of landmarks are sometimes used to help perform data association (see Figure 5.4). The *data-aligned rigidity-constrained exhaustive search (DARCES)* algorithm (Chen et al., 1999) is an example of a constellation-based data association method. The idea is that the distances between pairs of points in a constellation can be used as a type of unique identifier for data association.

Regardless of the type of data association employed, it is highly likely that if an estimation technique fails, the blame can be squarely placed on bad data association. For this reason, it is very important to acknowledge that misassociations will occur in practice, and therefore to design techniques to make the estimation problem robust to these occurrences. The next section, on outlier detection and rejection, will discuss some methods to help deal with this type of problem.

5.3 Handling Outliers

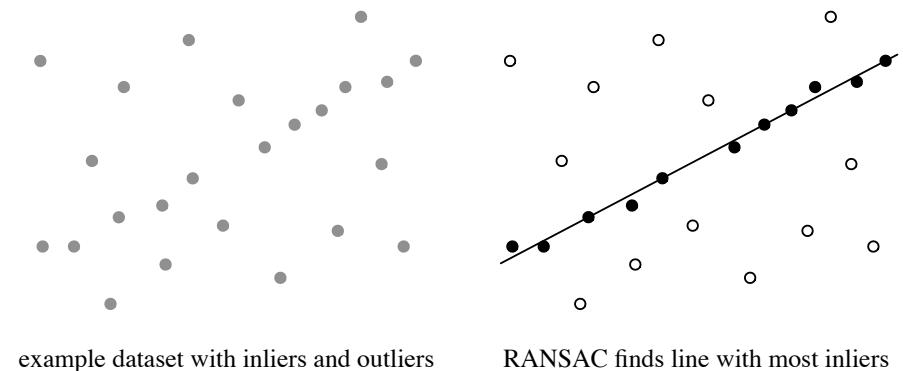
Data misassociation can certainly lead to an estimator completely diverging. However, data association is not the only cause of divergence. There are other factors that can cause any particular measurement to be very poor/incorrect. A classic example is multipath reflections of GPS timing signals near tall buildings. Figure 5.5 illustrates this point. A reflected signal can give a range measurement that is too long. In the absence of additional information, when the line-of-sight path is blocked the receiver has no way of knowing the longer path is incorrect.

We call measurements that are extremely improbable (according to our measurement model), *outliers*. Just how improbable is a matter of choice, but a common approach (in one dimensional data) is to consider measurements that are more than three standard deviations away from the mean to be outliers.

If we accept that a portion (possibly large) of our measurements

Figure 5.6

Line-fitting example. If a line is fit to all the data, the outliers will have a large impact on the result. The *random sample consensus* (*RANSAC*) approach is to classify the data as either an inlier or an outlier and then only use the inliers in the line fit.



could be outliers, we need to devise a means to detect and reduce/remove the influence of outliers on our estimators. We will discuss the two most common techniques to handle outliers:

- (i) Random sample consensus (Fischler and Bolles, 1981)
- (ii) M-Estimation (Zhang, 1997)

These can be used separately or in tandem. We will also touch on *adaptive estimation* (i.e., covariance estimation) and its connection to M-estimation.

5.3.1 RANSAC

Random sample consensus (*RANSAC*) is an iterative method to fit a parameterized model to a set of observed data containing outliers. *Outliers* are measurements that do not ‘fit’ a model, while *inliers* do ‘fit’. *RANSAC* is a probabilistic algorithm in the sense that its ability to find a reasonable answer can only be guaranteed to occur with a certain probability that improves with more time spent in the search. Figure 5.6 provides a classic line-fitting example in the presence of outliers.

RANSAC proceeds in an iterative manner. In the basic version, each iteration consists of the following five steps:

1. Select a (small) random subset of the original data to be hypothesized inliers (e.g., pick two points if fitting a line to xy -data).
2. Fit a model to the hypothesized inliers (e.g., a line is fit to two points).
3. Test the rest of the original data against the fitted model and classify as either inliers or outliers. If too few inliers are found, the iteration is labelled invalid and aborted.
4. Refit the model using both the hypothesized and classified inliers.
5. Evaluate the refit model in terms of the residual error of all the inlier data.

This is repeated for a large number of iterations, and the hypothesis with the lowest residual error is selected as the best.

A critical question to ask is how many iterations, k , are needed to ensure a subset is selected comprised solely of inliers, with probability p ? In general, this is difficult to answer. However, if we assume that each measurement is selected independently, and each has probability w of being an inlier, then the following relation holds:

$$1 - p = (1 - w^n)^k, \quad (5.36)$$

where n is the number of data points in the random subset and k is the number of iterations. Solving for k gives

$$k = \frac{\ln(1 - p)}{\ln(1 - w^n)}. \quad (5.37)$$

In reality, this can be thought of as an upper bound, as the data points are typically selected sequentially, not independently. There can also be constraints between the data points that complicate the selection of random subsets.

5.3.2 M-Estimation

Many of our earlier estimation techniques were shown to be minimizing a sum-of-squared-error cost function. The trouble with sum-of-squared-error cost functions, is that they are highly sensitive to outliers. A single large outlier can exercise a huge influence on the estimate because it dominates the quadratic cost. *M-estimation*³ modifies the shape of the cost function so that outliers do not dominate the solution.

We have seen previously that our overall nonlinear MAP objective function (for batch estimation) can be written in the form

$$J(\mathbf{x}) = \frac{1}{2} \sum_{i=1}^N \mathbf{e}_i(\mathbf{x})^T \mathbf{W}_i^{-1} \mathbf{e}_i(\mathbf{x}), \quad (5.38)$$

which is quadratic. The gradient of this objective function is

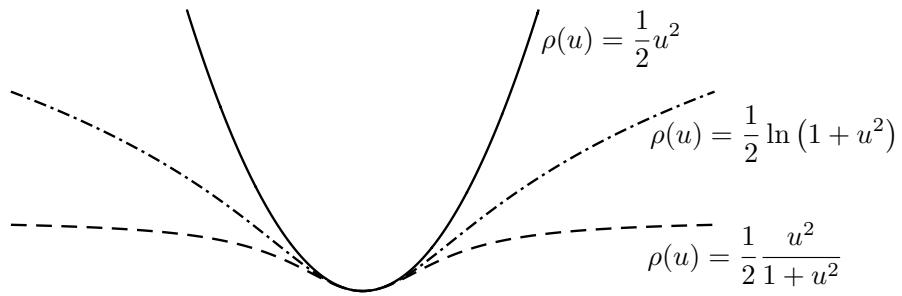
$$\frac{\partial J(\mathbf{x})}{\partial \mathbf{x}} = \sum_{i=1}^N \mathbf{e}_i(\mathbf{x})^T \mathbf{W}_i^{-1} \frac{\partial \mathbf{e}_i(\mathbf{x})}{\partial \mathbf{x}}, \quad (5.39)$$

which will be zero at a minimum. Let us now generalize this objective function and write it as

$$J'(\mathbf{x}) = \sum_{i=1}^N \alpha_i \rho(u_i(\mathbf{x})), \quad (5.40)$$

³ ‘M’ stands for ‘maximum likelihood-type’, i.e., a generalization of maximum likelihood (which we saw earlier was equivalent to the least-squares solution).

Figure 5.7
Comparison of quadratic, Cauchy, and Geman-McClure cost functions for scalar inputs.



where $\alpha_i > 0$ is a scalar weight,

$$u_i(\mathbf{x}) = \sqrt{\mathbf{e}_i(\mathbf{x})^T \mathbf{W}_i^{-1} \mathbf{e}_i(\mathbf{x})}, \quad (5.41)$$

and $\rho(u)$ is some nonlinear cost function; assume it is bounded, has a unique zero at $u = 0$, and increases monotonically with $u > 0$.

There are many possible cost functions, including

$$\underbrace{\rho(u) = \frac{1}{2}u^2}_{\text{quadratic}}, \quad \underbrace{\rho(u) = \frac{1}{2}\ln(1+u^2)}_{\text{Cauchy}}, \quad \underbrace{\rho(u) = \frac{1}{2}\frac{u^2}{1+u^2}}_{\text{Geman-McClure}}. \quad (5.42)$$

We refer to those that increase more slowly than quadratic as *robust cost functions*. This means that large errors will not carry as much weight and have little power on the solution due to a reduced gradient. Figure 5.7 depicts these options. Refer to Zhang (1997) for a more complete list of cost functions and MacTavish and Barfoot (2015) for a comparison study.

The gradient of our new objective function is simply

$$\frac{\partial J'(\mathbf{x})}{\partial \mathbf{x}} = \sum_{i=1}^N \alpha_i \frac{\partial \rho}{\partial u_i} \frac{\partial u_i}{\partial \mathbf{e}_i} \frac{\partial \mathbf{e}_i}{\partial \mathbf{x}}, \quad (5.43)$$

using the chain rule. Again, we want the gradient to go to zero if we are seeking a minimum. Substituting

$$\frac{\partial u_i}{\partial \mathbf{e}_i} = \frac{1}{u_i(\mathbf{x})} \mathbf{e}_i(\mathbf{x})^T \mathbf{W}_i^{-1}, \quad (5.44)$$

the gradient can be written as

$$\frac{\partial J'(\mathbf{x})}{\partial \mathbf{x}} = \sum_{i=1}^N \mathbf{e}_i(\mathbf{x})^T \mathbf{Y}_i(\mathbf{x})^{-1} \frac{\partial \mathbf{e}_i(\mathbf{x})}{\partial \mathbf{x}}, \quad (5.45)$$

where

$$\mathbf{Y}_i(\mathbf{x})^{-1} = \left. \frac{\alpha_i}{u_i(\mathbf{x})} \frac{\partial \rho}{\partial u_i} \right|_{u_i(\mathbf{x})} \mathbf{W}_i^{-1}, \quad (5.46)$$

is a new (inverse) covariance matrix that depends on \mathbf{x} ; we see that (5.45) is identical to (5.39), except that \mathbf{W}_i is now replaced with $\mathbf{Y}_i(\mathbf{x})$.

We are already using an iterative optimizer due to the nonlinear dependence of $\mathbf{e}_i(\mathbf{x})$ on \mathbf{x} , so it makes sense to evaluate $\mathbf{Y}_i(\mathbf{x})$ at the value of the state from the previous iteration, \mathbf{x}_{op} . This means we can simply work with the cost function

$$J''(\mathbf{x}) = \frac{1}{2} \sum_{i=1}^N \mathbf{e}_i(\mathbf{x})^T \mathbf{Y}_i(\mathbf{x}_{\text{op}})^{-1} \mathbf{e}_i(\mathbf{x}), \quad (5.47)$$

where

$$\mathbf{Y}_i(\mathbf{x}_{\text{op}})^{-1} = \frac{\alpha_i}{u_i(\mathbf{x}_{\text{op}})} \left. \frac{\partial \rho}{\partial u_i} \right|_{u_i(\mathbf{x}_{\text{op}})} \mathbf{W}_i^{-1}. \quad (5.48)$$

At each iteration, we solve the original least-squares problem, but with a modified covariance matrix that updates as \mathbf{x}_{op} updates. This is referred to as *iteratively reweighted least squares (IRLS)* (Holland and Welsch, 1977).

To see why this iterative scheme works, we can examine the gradient of $J''(\mathbf{x})$:

$$\frac{\partial J''(\mathbf{x})}{\partial \mathbf{x}} = \sum_{i=1}^N \mathbf{e}_i(\mathbf{x})^T \mathbf{Y}_i(\mathbf{x}_{\text{op}})^{-1} \frac{\partial \mathbf{e}_i(\mathbf{x})}{\partial \mathbf{x}}. \quad (5.49)$$

If the iterative scheme converges, we will have $\hat{\mathbf{x}} = \mathbf{x}_{\text{op}}$, so

$$\left. \frac{\partial J'(\mathbf{x})}{\partial \mathbf{x}} \right|_{\hat{\mathbf{x}}} = \left. \frac{\partial J''(\mathbf{x})}{\partial \mathbf{x}} \right|_{\hat{\mathbf{x}}} = \mathbf{0}, \quad (5.50)$$

and thus the two systems will have the same minimum (or minima). To be clear, however, the path taken to get to the optimum will differ if we minimize $J''(\mathbf{x})$ rather than $J'(\mathbf{x})$.

As an example, consider the case of the Cauchy robust cost function described above. The objective function becomes

$$J'(\mathbf{x}) = \frac{1}{2} \sum_{i=1}^N \alpha_i \ln (1 + \mathbf{e}_i(\mathbf{x})^T \mathbf{W}_i^{-1} \mathbf{e}_i(\mathbf{x})), \quad (5.51)$$

and we have

$$\mathbf{Y}_i(\mathbf{x}_{\text{op}})^{-1} = \frac{\alpha_i}{u_i(\mathbf{x}_{\text{op}})} \left. \frac{\partial \rho}{\partial u_i} \right|_{u_i(\mathbf{x}_{\text{op}})} \mathbf{W}_i^{-1} = \frac{\alpha_i}{u_i(\mathbf{x}_{\text{op}})} \frac{u_i(\mathbf{x}_{\text{op}})}{1 + u_i(\mathbf{x}_{\text{op}})^2} \mathbf{W}_i^{-1}, \quad (5.52)$$

and so

$$\mathbf{Y}_i(\mathbf{x}_{\text{op}}) = \frac{1}{\alpha_i} (1 + \mathbf{e}_i(\mathbf{x}_{\text{op}})^T \mathbf{W}_i^{-1} \mathbf{e}_i(\mathbf{x}_{\text{op}})) \mathbf{W}_i, \quad (5.53)$$

which we see is just an inflated version of the original (non-robust) covariance matrix, \mathbf{W}_i ; it gets bigger as the quadratic error, namely,

$\mathbf{e}_i(\mathbf{x}_{\text{op}})^T \mathbf{W}_i^{-1} \mathbf{e}_i(\mathbf{x}_{\text{op}})$, gets bigger. This makes sense because less trust is being assigned to cost terms that are very large (i.e., they are outliers).

5.3.3 Covariance Estimation

In the MAP estimation we have discussed so far, we have been dealing with cost functions of the form

$$J(\mathbf{x}) = \frac{1}{2} \sum_{i=1}^N \mathbf{e}_i(\mathbf{x})^T \mathbf{W}_i^{-1} \mathbf{e}_i(\mathbf{x}), \quad (5.54)$$

where we have assumed that the covariance associated with the inputs and measurements, \mathbf{W}_i , is known. We could try to determine this from some training data, where we have groundtruth for the state, but often this is not possible and so we resort to tuning by trial and error. This is partly why robust cost functions, as discussed in the last section, are necessary: our noise models are just not that good.

Another possibility is to try to estimate the covariances along with the state, which is sometimes called *adaptive estimation*. We can modify our MAP estimation problem to be

$$\{\hat{\mathbf{x}}, \hat{\mathbf{M}}\} = \arg \min_{\{\mathbf{x}, \mathbf{M}\}} J'(\mathbf{x}, \mathbf{M}), \quad (5.55)$$

where $\mathbf{M} = \{\mathbf{M}_1, \dots, \mathbf{M}_N\}$ is a convenient way to denote all of the unknown covariance matrices. This is similar to the idea of estimating a bias, as discussed earlier in this chapter; we simply include \mathbf{M}_i in the set of variables to be estimated. To make this work, we need to provide the estimator some guidance in the form of a prior over the possible values that \mathbf{M}_i is likely to have. Without a prior, the estimator can overfit to the data.

One possible prior is to assume that the covariance is distributed according to the *inverse-Wishart distribution*, which is defined over the real-valued, positive-definite matrices:

$$\mathbf{M}_i \sim \mathcal{W}^{-1}(\boldsymbol{\Psi}_i, \nu_i), \quad (5.56)$$

where $\boldsymbol{\Psi}_i > 0$ is called the *scale matrix*, $\nu_i > M_i - 1$ is the *degrees-of-freedom* parameter, and $M_i = \dim \mathbf{M}_i$. The inverse-Wishart PDF has the form

$$p(\mathbf{M}_i) = \frac{\det(\boldsymbol{\Psi}_i)^{\frac{\nu_i}{2}}}{2^{\frac{\nu_i M_i}{2}} \Gamma_{M_i}(\frac{\nu_i}{2})} \det(\mathbf{M}_i)^{-\frac{\nu_i + M_i + 1}{2}} \exp\left(-\frac{1}{2} \text{tr}(\boldsymbol{\Psi}_i \mathbf{M}_i^{-1})\right), \quad (5.57)$$

where $\Gamma_{M_i}(\cdot)$ is the *multivariate Gamma function*.

Under the MAP paradigm, the objective function is

$$J'(\mathbf{x}, \mathbf{M}) = -\ln p(\mathbf{x}, \mathbf{M} | \mathbf{z}), \quad (5.58)$$

where $\mathbf{z} = (\mathbf{z}_1, \dots, \mathbf{z}_N)$ represents all of our input and measurement data, respectively. Factoring the posterior,

$$p(\mathbf{x}, \mathbf{M} | \mathbf{z}) = p(\mathbf{x} | \mathbf{z}, \mathbf{M}) p(\mathbf{M}) = \prod_{i=1}^N p(\mathbf{x} | \mathbf{z}_i, \mathbf{M}_i) p(\mathbf{M}_i), \quad (5.59)$$

and plugging in the inverse-Wishart PDF, the objective function becomes

$$J'(\mathbf{x}, \mathbf{M}) = \frac{1}{2} \sum_{i=1}^N (\mathbf{e}_i(\mathbf{x})^T \mathbf{M}_i^{-1} \mathbf{e}_i(\mathbf{x}) - \alpha_i \ln(\det(\mathbf{M}_i^{-1})) + \text{tr}(\Psi_i \mathbf{M}_i^{-1})), \quad (5.60)$$

with $\alpha_i = \nu_i + M_i + 2$. We have dropped terms that do not depend on \mathbf{x} or \mathbf{M} .

Our strategy⁴ will be to first find the optimal \mathbf{M}_i (in terms of \mathbf{x}) so that we can eliminate it from the expression altogether. We do this by setting the derivative of $J'(\mathbf{x}, \mathbf{M})$ with respect to \mathbf{M}_i^{-1} to zero, since it is \mathbf{M}_i^{-1} that appears in the expression. Using some fairly standard matrix identities, we have

$$\frac{\partial J'(\mathbf{x}, \mathbf{M})}{\partial \mathbf{M}_i^{-1}} = \frac{1}{2} \mathbf{e}_i(\mathbf{x}) \mathbf{e}_i(\mathbf{x})^T - \frac{1}{2} \alpha_i \mathbf{M}_i + \frac{1}{2} \Psi_i. \quad (5.61)$$

Setting this to zero for a critical point and solving for \mathbf{M}_i , we have

$$\mathbf{M}_i(\mathbf{x}) = \underbrace{\frac{1}{\alpha_i} \Psi_i}_{\text{constant}} + \underbrace{\frac{1}{\alpha_i} \mathbf{e}_i(\mathbf{x}) \mathbf{e}_i(\mathbf{x})^T}_{\text{inflation}}, \quad (5.62)$$

which is quite an interesting expression. It shows that the optimal covariance, $\mathbf{M}_i(\mathbf{x})$, will be inflated from a constant wherever the residual errors in the trajectory estimate, $\mathbf{e}_i(\mathbf{x})$, are large. This inflated expression is similar to the IRLS covariance in (5.53) from the last section on M-estimation, implying a connection.

We can strengthen the connection to M-estimation by plugging the expression for the optimal $\mathbf{M}_i(\mathbf{x})$ back into $J'(\mathbf{x}, \mathbf{M})$, thereby eliminating it from the cost function. The resulting expression (after dropping factors that do not depend on \mathbf{x}) is

$$J'(\mathbf{x}) = \frac{1}{2} \sum_{i=1}^N \alpha_i \ln(1 + \mathbf{e}_i(\mathbf{x})^T \Psi_i^{-1} \mathbf{e}_i(\mathbf{x})). \quad (5.63)$$

This is exactly the form of a weighted Cauchy robust cost function discussed in the last section when the scale matrix is chosen to be our usual (non-robust) covariance: $\Psi_i = \mathbf{W}_i$. Thus, the original problem

⁴ An alternate strategy is to marginalize out \mathbf{M} from $p(\mathbf{x}, \mathbf{M} | \mathbf{z})$ from the beginning, which arrives at the same Cauchy-like final expression (Peretroukhin et al., 2016).

defined in (5.55) can be implemented as an M-estimation problem using IRLS.

It is not the case that the inflated covariance, $\mathbf{M}_i(\mathbf{x})$, from (5.62) is exactly the same as $\mathbf{Y}_i(\mathbf{x})$ from (5.53), although they are similar. This is because $\mathbf{M}_i(\mathbf{x})$ is the exact covariance needed to minimize our desired objective function, $J'(\mathbf{x})$, whereas $\mathbf{Y}_i(\mathbf{x})$ is an approximation used in our iterative scheme to minimize $J''(\mathbf{x})$. At convergence (i.e., when $\hat{\mathbf{x}} = \mathbf{x}_{\text{op}}$), the two methods have the same gradient and therefore the same minima, although they get there by slightly different paths.

It is quite appealing that in this specific case, the covariance estimation approach results in an equivalent M-estimation problem. It shows that the robust estimation approach can be explained from a MAP perspective, rather than simply being an ad hoc patch. It may be the case that this holds more generally, that all robust cost functions result from a particular choice of prior distribution over covariance matrices, $p(\mathbf{M})$. We leave it to the reader to investigate this further.

5.4 Summary

The main take-away points from this chapter are as follows:

1. There are always non-idealities (e.g., biases, outliers) that make the real estimation problem different from the clean mathematical setups discussed in this book. Sometimes these deviations result in performance reductions that are the main source of error in practice.
2. In some situations, we can fold the estimation of a bias into our estimation framework, and in others we cannot. This comes down to the question of observability.
3. In most practical estimation problems, outliers are a reality, and thus using some form of preprocessing (e.g., RANSAC) as well as a robust cost function that downplays the effect of outliers is a necessity.

The next part of the book will introduce techniques for handling state estimation in a three-dimensional world where objects are free to translate and rotate.

5.5 Exercises

5.5.1 Consider the discrete-time system

$$\begin{aligned} x_k &= x_{k-1} + v_k + \bar{v}, \\ d_k &= x_k, \end{aligned}$$

where \bar{v} is an unknown input bias. Set up the augmented-state system and determine if this system is observable.

5.5.2 Consider the discrete-time system

$$\begin{aligned}x_k &= x_{k-1} + v_{k-1}, \\v_k &= v_{k-1} + a_k, \\d_{1,k} &= x_k, \\d_{2,k} &= x_k + \bar{d},\end{aligned}$$

where \bar{d} is an unknown input bias (on just one of the two measurement equations). Set up the augmented-state system and determine if this system is observable.

5.5.3 How many RANSAC iterations, k , would be needed to pick a set of $n = 3$ inlier points with probability $p = 0.999$, given that each point has probability $w = 0.1$ of being an inlier?

5.5.4 What advantage might the robust cost function,

$$\rho(u) = \begin{cases} \frac{1}{2}u^2 & u^2 \leq 1 \\ \frac{2u^2}{1+u^2} - \frac{1}{2} & u^2 \geq 1 \end{cases},$$

have over the Geman-McClure cost function?

Part II

Three-Dimensional Machinery

6

Primer on Three-Dimensional Geometry

This chapter will introduce three-dimensional geometry and specifically the concept of a *rotation* and some of its representations. It pays particular attention to the establishment of *reference frames*. Sastry (1999) is a comprehensive reference on control for robotics that includes a background on three-dimensional geometry. Hughes (1986) also provides a good first-principles background.

6.1 Vectors and Reference Frames

Vehicles (e.g., robots, satellites, aircraft) are typically free to translate and rotate. Mathematically, they have six degrees of freedom: three in translation and three in rotation. This six-degree-of-freedom geometric configuration is known as the *pose* (position and orientation) of the vehicle. Some vehicles may have multiple bodies connected together; in this case each body has its own pose. We will consider only the single-body case here.

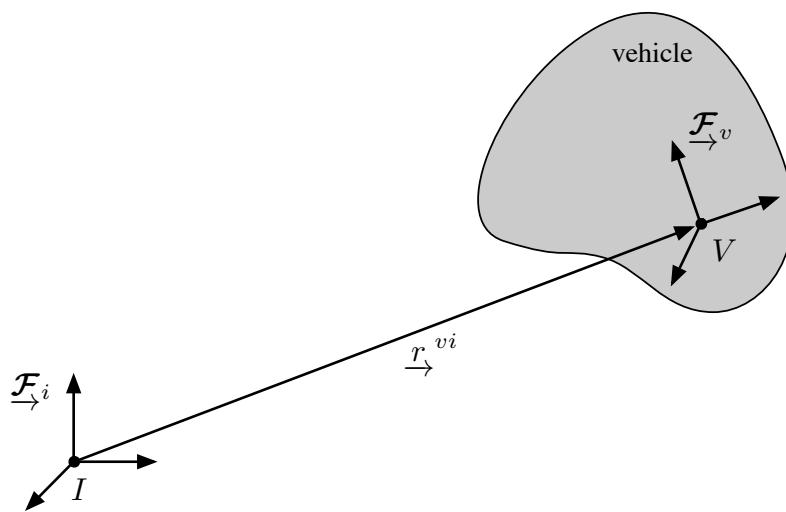
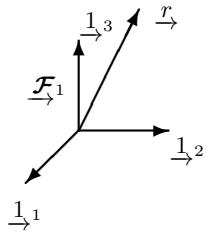


Figure 6.1
Vehicle and typical reference frames.

6.1.1 Reference Frames

The position of a point on a vehicle can be described with a vector, \underline{r}^{vi} , consisting of three components. Rotational motion is described by expressing the orientation of a reference frame on the vehicle, $\underline{\mathcal{F}}_v$, with respect to another frame, $\underline{\mathcal{F}}_i$. Figure 6.1 shows the typical setup for a single-body vehicle.

We will take a *vector* to be a quantity \underline{r} having length and direction. This vector can be expressed in a reference frame as



$$\begin{aligned}\underline{r} &= r_1 \underline{l}_1 + r_2 \underline{l}_2 + r_3 \underline{l}_3 \\ &= [r_1 \ r_2 \ r_3] \begin{bmatrix} \underline{l}_1 \\ \underline{l}_2 \\ \underline{l}_3 \end{bmatrix} \\ &= \mathbf{r}_1^T \underline{\mathcal{F}}_1.\end{aligned}\quad (6.1)$$

The quantity

$$\underline{\mathcal{F}}_1 = \begin{bmatrix} \underline{l}_1 \\ \underline{l}_2 \\ \underline{l}_3 \end{bmatrix}$$

is a column containing the basis vectors forming the reference frame $\underline{\mathcal{F}}_1$; we will always use basis vectors that are unit length, orthogonal, and arranged in a dextral (right-handed) fashion. We shall refer to $\underline{\mathcal{F}}_1$ as a *vectrix* (Hughes, 1986). The quantity

$$\mathbf{r}_1 = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix}$$

is a column matrix containing the *components* or *coordinates* of \underline{r} in reference frame $\underline{\mathcal{F}}_1$.

The vector can also be written as

$$\begin{aligned}\underline{r} &= [\underline{l}_1 \ \underline{l}_2 \ \underline{l}_3] \begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} \\ &= \underline{\mathcal{F}}_1^T \mathbf{r}_1.\end{aligned}$$

6.1.2 Dot Product

Consider two vectors, \underline{r} and \underline{s} , expressed in the same reference frame $\underline{\mathcal{F}}_1$:

$$\underline{r} = [r_1 \ r_2 \ r_3] \begin{bmatrix} \underline{l}_1 \\ \underline{l}_2 \\ \underline{l}_3 \end{bmatrix}, \quad \underline{s} = [\underline{l}_1 \ \underline{l}_2 \ \underline{l}_3] \begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix}.$$

The *dot product* (a.k.a., inner product) is given by

$$\begin{aligned}\underline{r} \cdot \underline{s} &= [r_1 \ r_2 \ r_3] \begin{bmatrix} \underline{1}_1 \\ \underline{1}_2 \\ \underline{1}_3 \end{bmatrix} \cdot \begin{bmatrix} \underline{1}_1 & \underline{1}_2 & \underline{1}_3 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix} \\ &= [r_1 \ r_2 \ r_3] \begin{bmatrix} \underline{1}_1 \cdot \underline{1}_1 & \underline{1}_1 \cdot \underline{1}_2 & \underline{1}_1 \cdot \underline{1}_3 \\ \underline{1}_2 \cdot \underline{1}_1 & \underline{1}_2 \cdot \underline{1}_2 & \underline{1}_2 \cdot \underline{1}_3 \\ \underline{1}_3 \cdot \underline{1}_1 & \underline{1}_3 \cdot \underline{1}_2 & \underline{1}_3 \cdot \underline{1}_3 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix}.\end{aligned}$$

But

$$\underline{1}_1 \cdot \underline{1}_1 = \underline{1}_2 \cdot \underline{1}_2 = \underline{1}_3 \cdot \underline{1}_3 = 1,$$

and

$$\underline{1}_1 \cdot \underline{1}_2 = \underline{1}_2 \cdot \underline{1}_3 = \underline{1}_3 \cdot \underline{1}_1 = 0.$$

Therefore,

$$\underline{r} \cdot \underline{s} = \mathbf{r}_1^T \mathbf{1} \mathbf{s}_1 = \mathbf{r}_1^T \mathbf{s}_1 = r_1 s_1 + r_2 s_2 + r_3 s_3.$$

The notation **1** will be used to designate the *identity matrix*. Its dimension can be inferred from context.

6.1.3 Cross Product

The *cross product* of two vectors expressed in the same reference frame is given by

$$\begin{aligned}\underline{r} \times \underline{s} &= [r_1 \ r_2 \ r_3] \begin{bmatrix} \underline{1}_1 \times \underline{1}_1 & \underline{1}_1 \times \underline{1}_2 & \underline{1}_1 \times \underline{1}_3 \\ \underline{1}_2 \times \underline{1}_1 & \underline{1}_2 \times \underline{1}_2 & \underline{1}_2 \times \underline{1}_3 \\ \underline{1}_3 \times \underline{1}_1 & \underline{1}_3 \times \underline{1}_2 & \underline{1}_3 \times \underline{1}_3 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix} \\ &= [r_1 \ r_2 \ r_3] \begin{bmatrix} 0 & \underline{1}_3 & -\underline{1}_2 \\ -\underline{1}_3 & 0 & \underline{1}_1 \\ \underline{1}_2 & -\underline{1}_1 & 0 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix} \\ &= [\underline{1}_1 \ \underline{1}_2 \ \underline{1}_3] \begin{bmatrix} 0 & -r_3 & r_2 \\ r_3 & 0 & -r_1 \\ -r_2 & r_1 & 0 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix} \\ &= \underline{\mathcal{F}}_1^T \mathbf{r}_1^* \mathbf{s}_1,\end{aligned}$$

where the fact that the basis vectors are orthogonal and arranged in a dextral fashion has been exploited. Hence, if \underline{r} and \underline{s} are expressed in the same reference frame, the 3×3 matrix

$$\mathbf{r}_1^* = \begin{bmatrix} 0 & -r_3 & r_2 \\ r_3 & 0 & -r_1 \\ -r_2 & r_1 & 0 \end{bmatrix}, \quad (6.2)$$

can be used to construct the components of the cross product. This matrix is *skew-symmetric*¹; that is,

$$(\mathbf{r}_1^\times)^T = -\mathbf{r}_1^\times.$$

It is easy to verify that

$$\mathbf{r}_1^\times \mathbf{r}_1 = \mathbf{0},$$

where $\mathbf{0}$ is a column matrix of zeros and

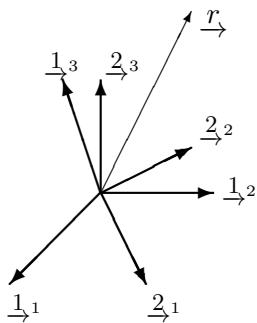
$$\mathbf{r}_1^\times \mathbf{s}_1 = -\mathbf{s}_1^\times \mathbf{r}_1.$$

6.2 Rotations

Critical to our ability to estimate how objects are moving in the world is the ability to parameterize the orientation, or rotation, of those objects. We begin by introducing rotation matrices and then provide some alternative representations.

6.2.1 Rotation Matrices

Let us consider two frames $\underline{\mathcal{F}}_1$ and $\underline{\mathcal{F}}_2$ with a common origin, and let us express \underline{r} in each frame:



$$\underline{r} = \underline{\mathcal{F}}_1^T \mathbf{r}_1 = \underline{\mathcal{F}}_2^T \mathbf{r}_2.$$

We seek to discover a relationship between the components in $\underline{\mathcal{F}}_1$, \mathbf{r}_1 , and those in $\underline{\mathcal{F}}_2$, \mathbf{r}_2 . We proceed as follows:

$$\begin{aligned} \underline{\mathcal{F}}_2^T \mathbf{r}_2 &= \underline{\mathcal{F}}_1^T \mathbf{r}_1, \\ \underline{\mathcal{F}}_2 \cdot \underline{\mathcal{F}}_2^T \mathbf{r}_2 &= \underline{\mathcal{F}}_2 \cdot \underline{\mathcal{F}}_1^T \mathbf{r}_1, \\ \mathbf{r}_2 &= \mathbf{C}_{21} \mathbf{r}_1. \end{aligned}$$

We have defined

$$\begin{aligned} \mathbf{C}_{21} &= \underline{\mathcal{F}}_2 \cdot \underline{\mathcal{F}}_1^T \\ &= \begin{bmatrix} \underline{2}_1 \\ \underline{2}_2 \\ \underline{2}_3 \end{bmatrix} \cdot \begin{bmatrix} \underline{1}_1 & \underline{1}_2 & \underline{1}_3 \end{bmatrix} \\ &= \begin{bmatrix} \underline{2}_1 \cdot \underline{1}_1 & \underline{2}_1 \cdot \underline{1}_2 & \underline{2}_1 \cdot \underline{1}_3 \\ \underline{2}_2 \cdot \underline{1}_1 & \underline{2}_2 \cdot \underline{1}_2 & \underline{2}_2 \cdot \underline{1}_3 \\ \underline{2}_3 \cdot \underline{1}_1 & \underline{2}_3 \cdot \underline{1}_2 & \underline{2}_3 \cdot \underline{1}_3 \end{bmatrix}. \end{aligned}$$

¹ There are many equivalent notations in the literature for this skew-symmetric definition: $\mathbf{r}_1^\times = \hat{\mathbf{r}}_1 = \mathbf{r}_1^\wedge = -[[\mathbf{r}_1]] = [\mathbf{r}_1]_\times$. For now, we use the first one, since it makes an obvious connection to the cross product; later we will also use $(\cdot)^\wedge$, as this is in common use in robotics.

The matrix \mathbf{C}_{21} is called a *rotation matrix*. It is sometimes referred to as a ‘direction cosine matrix’ since the dot product of two unit vectors is just the cosine of the angle between them.

The unit vectors in $\underline{\mathcal{F}}_2$ can be related to those in $\underline{\mathcal{F}}_1$:

$$\underline{\mathcal{F}}_1^T = \underline{\mathcal{F}}_2^T \mathbf{C}_{21}. \quad (6.3)$$

Rotation matrices possess some special properties:

$$\mathbf{r}_1 = \mathbf{C}_{21}^{-1} \mathbf{r}_2 = \mathbf{C}_{12} \mathbf{r}_2.$$

But, $\mathbf{C}_{21}^T = \mathbf{C}_{12}$. Hence,

$$\mathbf{C}_{12} = \mathbf{C}_{21}^{-1} = \mathbf{C}_{21}^T. \quad (6.4)$$

We say that \mathbf{C}_{21} is an *orthonormal matrix* because its inverse is equal to its transpose.

Consider three reference frames $\underline{\mathcal{F}}_1$, $\underline{\mathcal{F}}_2$, and $\underline{\mathcal{F}}_3$. The components of a vector \underline{r} in these three frames are \mathbf{r}_1 , \mathbf{r}_2 , and \mathbf{r}_3 . Now,

$$\mathbf{r}_3 = \mathbf{C}_{32} \mathbf{r}_2 = \mathbf{C}_{32} \mathbf{C}_{21} \mathbf{r}_1.$$

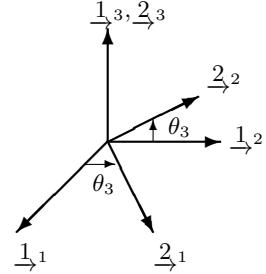
But, $\mathbf{r}_3 = \mathbf{C}_{31} \mathbf{r}_1$, and therefore

$$\mathbf{C}_{31} = \mathbf{C}_{32} \mathbf{C}_{21}.$$

6.2.2 Principal Rotations

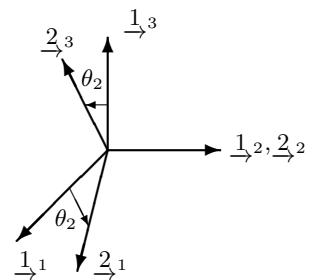
Before considering more general rotations, it is useful to consider rotations about one basis vector. The situation where $\underline{\mathcal{F}}_2$ has been rotated from $\underline{\mathcal{F}}_1$ through a rotation about the 3-axis is shown in the figure. The rotation matrix in this case is

$$\mathbf{C}_3 = \begin{bmatrix} \cos \theta_3 & \sin \theta_3 & 0 \\ -\sin \theta_3 & \cos \theta_3 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (6.5)$$

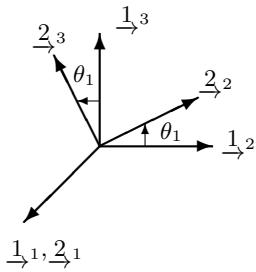


For a rotation about the 2-axis, the rotation matrix is

$$\mathbf{C}_2 = \begin{bmatrix} \cos \theta_2 & 0 & -\sin \theta_2 \\ 0 & 1 & 0 \\ \sin \theta_2 & 0 & \cos \theta_2 \end{bmatrix}. \quad (6.6)$$



For a rotation about the 1-axis, the rotation matrix is



$$\mathbf{C}_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta_1 & \sin \theta_1 \\ 0 & -\sin \theta_1 & \cos \theta_1 \end{bmatrix}. \quad (6.7)$$

6.2.3 Alternate Rotation Representations

We have seen one way of discussing the orientation of one reference frame with respect to another: the *rotation matrix*. The rotation matrix describes orientation both globally and uniquely. This requires nine parameters (they are not independent). There are a number of other alternatives.

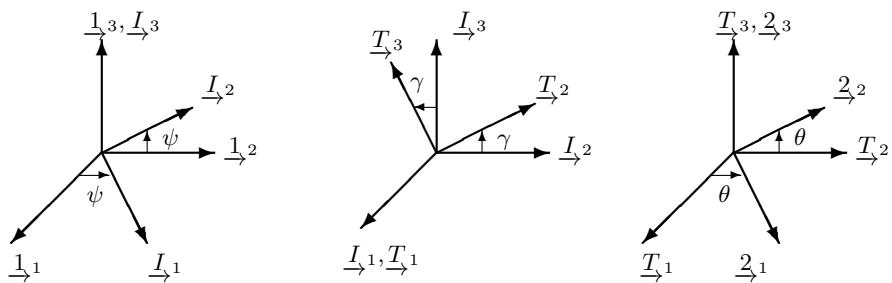
The key thing to realize about the different representations of rotations, is that there are always only three underlying degrees of freedom. The representations that have more than three parameters must have associated constraints to limit the number of degrees of freedom to three. The representations that have exactly three parameters have associated singularities. There is no perfect representation that is minimal (i.e., having only three parameters) and that is also free of singularities (Stuelpnagel, 1964).

Leonhard Euler (1707-1783) is considered to be the preeminent mathematician of the eighteenth century and one of the greatest mathematicians to have ever lived. He made important discoveries in fields as diverse as infinitesimal calculus and graph theory. He also introduced much of the modern mathematical terminology and notation, particularly for mathematical analysis, such as the notion of a mathematical function. He is also renowned for his work in mechanics, fluid dynamics, optics, astronomy, and music theory.

Euler Angles

The orientation of one reference frame with respect to another can also be specified by a sequence of three principal rotations. One possible sequence is as follows:

- (i) A rotation ψ about the original 3-axis
- (ii) A rotation γ about the intermediate 1-axis
- (iii) A rotation θ about the transformed 3-axis



This is called a 3-1-3 sequence and is the one originally used by Euler.

In the classical mechanics literature, the angles are referred to by the following names:

- θ : spin angle
- γ : nutation angle
- ψ : precession angle

The rotation matrix from frame 1 to frame 2 is given by

$$\begin{aligned}\mathbf{C}_{21}(\theta, \gamma, \psi) &= \mathbf{C}_{2T}\mathbf{C}_{TI}\mathbf{C}_{I1} \\ &= \mathbf{C}_3(\theta)\mathbf{C}_1(\gamma)\mathbf{C}_3(\psi) \\ &= \begin{bmatrix} c_\theta c_\psi - s_\theta c_\gamma s_\psi & s_\psi c_\theta + c_\gamma s_\theta c_\psi & s_\gamma s_\theta \\ -c_\psi s_\theta - c_\theta c_\gamma s_\psi & -s_\psi s_\theta + c_\theta c_\gamma c_\psi & s_\gamma c_\theta \\ s_\psi s_\gamma & -s_\gamma c_\psi & c_\gamma \end{bmatrix}. \quad (6.8)\end{aligned}$$

We have made the abbreviations $s = \sin$, $c = \cos$.

Another possible sequence that can be used is as follows:

- (i) A rotation θ_1 about the original 1-axis ('roll' rotation)
- (ii) A rotation θ_2 about the intermediate 2-axis ('pitch' rotation)
- (iii) A rotation θ_3 about the transformed 3-axis ('yaw' rotation)

This sequence, which is very common in aerospace applications, is called the 1-2-3 attitude sequence or the 'roll-pitch-yaw' convention. In this case, the rotation matrix from frame 1 to frame 2 is given by

$$\begin{aligned}\mathbf{C}_{21}(\theta_3, \theta_2, \theta_1) &= \mathbf{C}_3(\theta_3)\mathbf{C}_2(\theta_2)\mathbf{C}_1(\theta_1) \\ &= \begin{bmatrix} c_2 c_3 & c_1 s_3 + s_1 s_2 c_3 & s_1 s_3 - c_1 s_2 c_3 \\ -c_2 s_3 & c_1 c_3 - s_1 s_2 s_3 & s_1 c_3 + c_1 s_2 s_3 \\ s_2 & -s_1 c_2 & c_1 c_2 \end{bmatrix}, \quad (6.9)\end{aligned}$$

where $s_i = \sin \theta_i$, $c_i = \cos \theta_i$.

All Euler sequences have singularities. For instance, if $\gamma = 0$ for the 3-1-3 sequence, then the angles θ and ψ become associated with the same degree of freedom and cannot be uniquely determined.

For the 1-2-3 sequence, a singularity exists at $\theta_2 = \pi/2$. In this case,

$$\mathbf{C}_{21}(\theta_3, \frac{\pi}{2}, \theta_1) = \begin{bmatrix} 0 & \sin(\theta_1 + \theta_3) & -\cos(\theta_1 + \theta_3) \\ 0 & \cos(\theta_1 + \theta_3) & \sin(\theta_1 + \theta_3) \\ 1 & 0 & 0 \end{bmatrix}.$$

Therefore, θ_1 and θ_3 are associated with the same rotation. However, this is only a problem if we want to recover the rotation angles from the rotation matrix.

Infinitesimal Rotations

Consider the 1-2-3 transformation when the angles $\theta_1, \theta_2, \theta_3$ are small. In this case, we make the approximations $c_i \approx 1$, $s_i \approx \theta_i$ and neglect

products of small angles, $\theta_i \theta_j \approx 0$. Then we have

$$\begin{aligned}\mathbf{C}_{21} &\approx \begin{bmatrix} 1 & \theta_3 & -\theta_2 \\ -\theta_3 & 1 & \theta_1 \\ \theta_2 & -\theta_1 & 1 \end{bmatrix} \\ &\approx \mathbf{1} - \boldsymbol{\theta}^\times,\end{aligned}\quad (6.10)$$

where

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix},$$

which is referred to as a *rotation vector*.

It is easy to show that the form of the rotation matrix for infinitesimal rotations (i.e., ‘small angle approximation’) does not depend on the order in which the rotations are performed. For example, we can show that the same result is obtained for a 2-1-3 Euler sequence.

Euler Parameters

Euler’s rotation theorem says that the most general motion of a rigid body with one point fixed is a rotation about an axis through that point.

Let us denote the *axis of rotation* by $\mathbf{a} = [a_1 \ a_2 \ a_3]^T$ and assume that it is a unit vector:

$$\mathbf{a}^T \mathbf{a} = a_1^2 + a_2^2 + a_3^2 = 1. \quad (6.11)$$

The *angle of rotation* is ϕ . We state, without proof, that the rotation matrix in this case is given by

$$\mathbf{C}_{21} = \cos \phi \mathbf{1} + (1 - \cos \phi) \mathbf{a} \mathbf{a}^T - \sin \phi \mathbf{a}^\times. \quad (6.12)$$

It does not matter in which frame \mathbf{a} is expressed because

$$\mathbf{C}_{21} \mathbf{a} = \mathbf{a}. \quad (6.13)$$

The combination of variables,

$$\eta = \cos \frac{\phi}{2}, \quad \boldsymbol{\varepsilon} = \mathbf{a} \sin \frac{\phi}{2} = \begin{bmatrix} a_1 \sin(\phi/2) \\ a_2 \sin(\phi/2) \\ a_3 \sin(\phi/2) \end{bmatrix} = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \end{bmatrix}, \quad (6.14)$$

is particularly useful. The four parameters $\{\varepsilon, \eta\}$ are called the *Euler parameters* associated with a rotation². They are not independent because they satisfy the constraint

$$\eta^2 + \varepsilon_1^2 + \varepsilon_2^2 + \varepsilon_3^2 = 1.$$

² These are sometimes referred to as *unit-length quaternions* when stacked as $\mathbf{q} = \begin{bmatrix} \boldsymbol{\varepsilon} \\ \eta \end{bmatrix}$.

These are discussed in more detail below.

The rotation matrix can be expressed in terms of the Euler parameters as

$$\begin{aligned}\mathbf{C}_{21} &= (\eta^2 - \boldsymbol{\varepsilon}^T \boldsymbol{\varepsilon}) \mathbf{1} + 2\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^T - 2\eta\boldsymbol{\varepsilon}^\times \\ &= \begin{bmatrix} 1 - 2(\varepsilon_2^2 + \varepsilon_3^2) & 2(\varepsilon_1\varepsilon_2 + \varepsilon_3\eta) & 2(\varepsilon_1\varepsilon_3 - \varepsilon_2\eta) \\ 2(\varepsilon_2\varepsilon_1 - \varepsilon_3\eta) & 1 - 2(\varepsilon_3^2 + \varepsilon_1^2) & 2(\varepsilon_2\varepsilon_3 + \varepsilon_1\eta) \\ 2(\varepsilon_3\varepsilon_1 + \varepsilon_2\eta) & 2(\varepsilon_3\varepsilon_2 - \varepsilon_1\eta) & 1 - 2(\varepsilon_1^2 + \varepsilon_2^2) \end{bmatrix}. \quad (6.15)\end{aligned}$$

Euler parameters are useful in many spacecraft applications. There are no singularities associated with them, and the calculation of the rotation matrix does not involve trigonometric functions, which is a significant numerical advantage. The only drawback is the use of four parameters instead of three, as is the case with Euler angles; this makes it challenging to perform some estimation problems because the constraint must be enforced.

Quaternions

We will use the notation of Barfoot et al. (2011) for this section. A quaternion will be a 4×1 column that may be written as

$$\mathbf{q} = \begin{bmatrix} \boldsymbol{\varepsilon} \\ \eta \end{bmatrix}, \quad (6.16)$$

where $\boldsymbol{\varepsilon}$ is a 3×1 and η is a scalar. The quaternion left-hand compound operator, $+$, and the right-hand compound operator, \oplus , will be defined as

$$\mathbf{q}^+ = \begin{bmatrix} \eta \mathbf{1} - \boldsymbol{\varepsilon}^\times & \boldsymbol{\varepsilon} \\ -\boldsymbol{\varepsilon}^T & \eta \end{bmatrix}, \quad \mathbf{q}^\oplus = \begin{bmatrix} \eta \mathbf{1} + \boldsymbol{\varepsilon}^\times & \boldsymbol{\varepsilon} \\ -\boldsymbol{\varepsilon}^T & \eta \end{bmatrix}. \quad (6.17)$$

The inverse operator, -1 , will be defined by

$$\mathbf{q}^{-1} = \begin{bmatrix} -\boldsymbol{\varepsilon} \\ \eta \end{bmatrix}. \quad (6.18)$$

Let \mathbf{u} , \mathbf{v} , and \mathbf{w} be quaternions. Then some useful identities are

$$\mathbf{u}^+ \mathbf{v} \equiv \mathbf{v}^\oplus \mathbf{u}, \quad (6.19)$$

and

$$\begin{aligned}(\mathbf{u}^+)^T &\equiv (\mathbf{u}^+)^{-1} \equiv (\mathbf{u}^{-1})^+, & (\mathbf{u}^\oplus)^T &\equiv (\mathbf{u}^\oplus)^{-1} \equiv (\mathbf{u}^{-1})^\oplus, \\ (\mathbf{u}^+ \mathbf{v})^{-1} &\equiv \mathbf{v}^{-1+} \mathbf{u}^{-1}, & (\mathbf{u}^\oplus \mathbf{v})^{-1} &\equiv \mathbf{v}^{-1\oplus} \mathbf{u}^{-1}, \\ (\mathbf{u}^+ \mathbf{v})^+ \mathbf{w} &\equiv \mathbf{u}^+ (\mathbf{v}^+ \mathbf{w}) \equiv \mathbf{u}^+ \mathbf{v}^+ \mathbf{w}, & (\mathbf{u}^\oplus \mathbf{v})^\oplus \mathbf{w} &\equiv \mathbf{u}^\oplus (\mathbf{v}^\oplus \mathbf{w}) \equiv \mathbf{u}^\oplus \mathbf{v}^\oplus \mathbf{w}, \\ \alpha \mathbf{u}^+ + \beta \mathbf{v}^+ &\equiv (\alpha \mathbf{u} + \beta \mathbf{v})^+, & \alpha \mathbf{u}^\oplus + \beta \mathbf{v}^\oplus &\equiv (\alpha \mathbf{u} + \beta \mathbf{v})^\oplus, \end{aligned} \quad (6.20)$$

where α and β are scalars. We also have

$$\mathbf{u}^+ \mathbf{v}^\oplus \equiv \mathbf{v}^\oplus \mathbf{u}^+. \quad (6.21)$$

The proofs are left to the reader.

Quaternions were first described by Sir William Rowan Hamilton (1805–1865) in 1843 and applied to mechanics in three-dimensional space. Hamilton was an Irish physicist, astronomer, and mathematician, who made important contributions to classical mechanics, optics, and algebra. His studies of mechanical and optical systems led him to discover new mathematical concepts and techniques. His best known contribution to mathematical physics is the reformulation of Newtonian mechanics, now called Hamiltonian mechanics. This work has proven central to the modern study of classical field theories such as electromagnetism, and to the development of quantum mechanics. In pure mathematics, he is best known as the inventor of quaternions.

Quaternions form a *non-commutative group*³ under both the + and \oplus operations. Many of the identities above are prerequisites to showing this fact. The identity element of this group, $\boldsymbol{\iota} = [0 \ 0 \ 0 \ 1]^T$, is such that

$$\boldsymbol{\iota}^+ = \boldsymbol{\iota}^\oplus = \mathbf{1}, \quad (6.22)$$

where $\mathbf{1}$ is the 4×4 identity matrix.

Rotations may be represented in this notation by using a unit-length quaternion, \mathbf{q} , such that

$$\mathbf{q}^T \mathbf{q} = 1. \quad (6.23)$$

These form a *sub-group* that can be used to represent rotations.

To rotate a point (in homogeneous form)

$$\mathbf{v} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (6.24)$$

to another frame using the rotation, \mathbf{q} , we compute

$$\mathbf{u} = \mathbf{q}^+ \mathbf{v}^+ \mathbf{q}^{-1} = \mathbf{q}^+ \mathbf{q}^{-1\oplus} \mathbf{v} = \mathbf{R} \mathbf{v}, \quad (6.25)$$

where

$$\mathbf{R} = \mathbf{q}^+ \mathbf{q}^{-1\oplus} = \mathbf{q}^{-1\oplus} \mathbf{q}^+ = \mathbf{q}^{\oplus T} \mathbf{q}^+ = \begin{bmatrix} \mathbf{C} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix}, \quad (6.26)$$

and \mathbf{C} is the 3×3 rotation matrix with which we are now familiar. We have included various forms for \mathbf{R} to show the different structures this transformation can take.

Gibbs Vector

Yet another way that we can parameterize rotations is through the *Gibbs vector*. In terms of axis/angle parameters discussed earlier, the Gibbs vector, \mathbf{g} , is given by

$$\mathbf{g} = \mathbf{a} \tan \frac{\phi}{2}, \quad (6.27)$$

which we note has a singularity at $\phi = \pi$, so this parameterization does not work well for all angles. The rotation matrix, \mathbf{C} , can then be written in terms of the Gibbs vector as

$$\mathbf{C} = (\mathbf{1} + \mathbf{g}^\times)^{-1} (\mathbf{1} - \mathbf{g}^\times) = \frac{1}{1 + \mathbf{g}^T \mathbf{g}} ((1 - \mathbf{g}^T \mathbf{g}) \mathbf{1} + 2\mathbf{g}\mathbf{g}^T - 2\mathbf{g}^\times). \quad (6.28)$$

Josiah Willard Gibbs (1839-1903) was an American scientist who made important theoretical contributions to physics, chemistry, and mathematics. As a mathematician, he invented modern vector calculus (independently of the British scientist Oliver Heaviside, who carried out similar work during the same period). The Gibbs vector is also sometimes known as the *Cayley-Rodrigues parameters*.

³ The next chapter will discuss group theory as it pertains to rotations in much more detail.

Substituting in the Gibbs vector definition, the right-hand expression becomes

$$\mathbf{C} = \frac{1}{1 + \tan^2 \frac{\phi}{2}} \left(\left(1 - \tan^2 \frac{\phi}{2} \right) \mathbf{1} + 2 \tan^2 \frac{\phi}{2} \mathbf{a} \mathbf{a}^T - 2 \tan \frac{\phi}{2} \mathbf{a}^\times \right), \quad (6.29)$$

where we have used that $\mathbf{a}^T \mathbf{a} = 1$. Utilizing that $(1 + \tan^2 \frac{\phi}{2})^{-1} = \cos^2 \frac{\phi}{2}$, we have

$$\begin{aligned} \mathbf{C} &= \underbrace{\left(\cos^2 \frac{\phi}{2} - \sin^2 \frac{\phi}{2} \right)}_{\cos \phi} \mathbf{1} + \underbrace{2 \sin^2 \frac{\phi}{2}}_{1 - \cos \phi} \mathbf{a} \mathbf{a}^T - \underbrace{2 \sin \frac{\phi}{2} \cos \frac{\phi}{2}}_{\sin \phi} \mathbf{a}^\times \\ &= \cos \phi \mathbf{1} + (1 - \cos \phi) \mathbf{a} \mathbf{a}^T - \sin \phi \mathbf{a}^\times, \end{aligned} \quad (6.30)$$

which is our usual expression for the rotation matrix in terms of the axis/angle parameters.

To relate the two expressions for \mathbf{C} in terms of \mathbf{g} given in (6.28), we first note that

$$(\mathbf{1} + \mathbf{g}^\times)^{-1} = \mathbf{1} - \mathbf{g}^\times + \mathbf{g}^\times \mathbf{g}^\times - \mathbf{g}^\times \mathbf{g}^\times \mathbf{g}^\times + \cdots = \sum_{n=0}^{\infty} (-\mathbf{g}^\times)^n. \quad (6.31)$$

Then we observe that

$$\begin{aligned} \mathbf{g}^T \mathbf{g} (\mathbf{1} + \mathbf{g}^\times)^{-1} &= (\mathbf{g}^T \mathbf{g}) \mathbf{1} - \underbrace{(\mathbf{g}^T \mathbf{g}) \mathbf{g}^\times}_{-\mathbf{g}^\times \mathbf{g}^\times \mathbf{g}^\times} + \underbrace{(\mathbf{g}^T \mathbf{g}) \mathbf{g}^\times \mathbf{g}^\times}_{-\mathbf{g}^\times \mathbf{g}^\times \mathbf{g}^\times \mathbf{g}^\times} - \underbrace{(\mathbf{g}^T \mathbf{g}) \mathbf{g}^\times \mathbf{g}^\times \mathbf{g}^\times}_{-\mathbf{g}^\times \mathbf{g}^\times \mathbf{g}^\times \mathbf{g}^\times \mathbf{g}^\times} + \cdots \\ &= \mathbf{1} + \mathbf{g} \mathbf{g}^T - \mathbf{g}^\times - (\mathbf{1} + \mathbf{g}^\times)^{-1}, \end{aligned} \quad (6.32)$$

where we have used the following manipulation several times:

$$(\mathbf{g}^T \mathbf{g}) \mathbf{g}^\times = (-\mathbf{g}^\times \mathbf{g}^\times + \mathbf{g} \mathbf{g}^T) \mathbf{g}^\times = -\mathbf{g}^\times \mathbf{g}^\times \mathbf{g}^\times + \mathbf{g} \underbrace{\mathbf{g}^T \mathbf{g}^\times}_0 = -\mathbf{g}^\times \mathbf{g}^\times \mathbf{g}^\times. \quad (6.33)$$

Therefore we have that

$$(\mathbf{1} + \mathbf{g}^T \mathbf{g}) (\mathbf{1} + \mathbf{g}^\times)^{-1} = \mathbf{1} + \mathbf{g} \mathbf{g}^T - \mathbf{g}^\times, \quad (6.34)$$

and thus

$$\begin{aligned} &(\mathbf{1} + \mathbf{g}^T \mathbf{g}) \underbrace{(\mathbf{1} + \mathbf{g}^\times)^{-1}}_{\mathbf{C}} (\mathbf{1} - \mathbf{g}^\times) = (\mathbf{1} + \mathbf{g} \mathbf{g}^T - \mathbf{g}^\times) (\mathbf{1} - \mathbf{g}^\times) \\ &= \mathbf{1} + \mathbf{g} \mathbf{g}^T - 2 \mathbf{g}^\times - \mathbf{g} \underbrace{\mathbf{g}^T \mathbf{g}^\times}_0 + \underbrace{\mathbf{g}^\times \mathbf{g}^\times}_{-\mathbf{g}^T \mathbf{g} \mathbf{1} + \mathbf{g} \mathbf{g}^T} = (\mathbf{1} - \mathbf{g}^T \mathbf{g}) \mathbf{1} + 2 \mathbf{g} \mathbf{g}^T - 2 \mathbf{g}^\times. \end{aligned} \quad (6.35)$$

Dividing both sides by $(\mathbf{1} + \mathbf{g}^T \mathbf{g})$ provides the desired result.

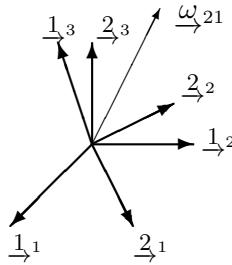
6.2.4 Rotational Kinematics

In the last section, we showed that the orientation of one frame $\underline{\mathcal{F}}_2$ with respect to another $\underline{\mathcal{F}}_1$ could be parameterized in different ways. In other words, the rotation matrix could be written as a function of Euler angles or Euler parameters. However, in most applications the orientation changes with time and thus we must introduce the vehicle *kinematics*, which form an important part of the vehicle's motion model.

We will first introduce the concept of angular velocity, then acceleration in a rotating frame. We will finish with expressions that relate the rate of change of the orientation parameterization to angular velocity.

Angular Velocity

Let frame $\underline{\mathcal{F}}_2$ rotate with respect to frame $\underline{\mathcal{F}}_1$. The angular velocity of frame 2 with respect to frame 1 is denoted by $\underline{\omega}_{21}$. The angular velocity of frame 1 with respect to 2 is $\underline{\omega}_{12} = -\underline{\omega}_{21}$.



The magnitude of $\underline{\omega}_{21}$, $|\underline{\omega}_{21}| = \sqrt{(\underline{\omega}_{21} \cdot \underline{\omega}_{21})}$, is the rate of rotation. The direction of $\underline{\omega}_{21}$ (i.e., the unit vector in the direction of $\underline{\omega}_{21}$, which is $|\underline{\omega}_{21}|^{-1} \underline{\omega}_{21}$) is the *instantaneous axis of rotation*.

Observers in the frames $\underline{\mathcal{F}}_2$ and $\underline{\mathcal{F}}_1$ do not see the same motion because of their own relative motions. Let us denote the *vector time derivative* as seen in $\underline{\mathcal{F}}_1$ by $(\cdot)^\bullet$ and that seen in $\underline{\mathcal{F}}_2$ by $(\cdot)^\circ$. Therefore,

$$\underline{\mathcal{F}}_1^\bullet = \underline{0}, \quad \underline{\mathcal{F}}_2^\circ = \underline{0}.$$

It can be shown that

$$\underline{2}_1^\bullet = \underline{\omega}_{21} \times \underline{2}_1, \quad \underline{2}_2^\bullet = \underline{\omega}_{21} \times \underline{2}_2, \quad \underline{2}_3^\bullet = \underline{\omega}_{21} \times \underline{2}_3,$$

or equivalently

$$\begin{bmatrix} \underline{2}_1^\bullet & \underline{2}_2^\bullet & \underline{2}_3^\bullet \end{bmatrix} = \underline{\omega}_{21} \times \begin{bmatrix} \underline{2}_1 & \underline{2}_2 & \underline{2}_3 \end{bmatrix},$$

or

$$\underline{\mathcal{F}}_2^{\bullet T} = \underline{\omega}_{21} \times \underline{\mathcal{F}}_2^T. \quad (6.36)$$

We want to determine the time derivative of an arbitrary vector expressed in both frames:

$$\dot{\underline{r}} = \underline{\mathcal{F}}_1^T \underline{r}_1 = \underline{\mathcal{F}}_2^T \underline{r}_2.$$

Therefore, the time derivative as seen in $\underline{\mathcal{F}}_1$ is

$$\dot{\underline{r}}^\bullet = \underline{\mathcal{F}}_1^T \underline{r}_1 + \underline{\mathcal{F}}_1^T \dot{\underline{r}}_1 = \underline{\mathcal{F}}_1^T \dot{\underline{r}}_1. \quad (6.37)$$

In a similar way,

$$\dot{\underline{r}}^\circ = \underline{\mathcal{F}}_2^{\circ T} \underline{r}_2 + \underline{\mathcal{F}}_2^T \dot{\underline{r}}_2 = \underline{\mathcal{F}}_2^T \dot{\underline{r}}_2 = \underline{\mathcal{F}}_2^T \dot{\underline{r}}_2. \quad (6.38)$$

(Note that for nonvectors, $(\cdot)^\circ = (\cdot)$, i.e., $\dot{\underline{r}}_2^\circ = \dot{\underline{r}}_2$.)

Alternatively, the time derivative as seen in $\underline{\mathcal{F}}_1$, but expressed in $\underline{\mathcal{F}}_2$, is

$$\begin{aligned} \dot{\underline{r}}^\bullet &= \underline{\mathcal{F}}_2^T \dot{\underline{r}}_2 + \underline{\mathcal{F}}_2^T \underline{r}_2 \\ &= \underline{\mathcal{F}}_2^T \dot{\underline{r}}_2 + \underline{\omega}_{21} \times \underline{\mathcal{F}}_2^T \underline{r}_2 \\ &= \dot{\underline{r}}^\circ + \underline{\omega}_{21} \times \dot{\underline{r}}. \end{aligned} \quad (6.39)$$

The above is true for any vector $\dot{\underline{r}}$. The most important application occurs when $\dot{\underline{r}}$ denotes position, $\underline{\mathcal{F}}_1$ is a nonrotating inertial reference frame, and $\underline{\mathcal{F}}_2$ is a frame that rotates with a body, vehicle, etc. In this case, (6.39) expresses the velocity in the inertial frame in terms of the motion in the second frame.

Now, express the angular velocity in $\underline{\mathcal{F}}_2$:

$$\underline{\omega}_{21} = \underline{\mathcal{F}}_2^T \underline{\omega}_2^{21}. \quad (6.40)$$

Therefore,

$$\begin{aligned} \dot{\underline{r}}^\bullet &= \underline{\mathcal{F}}_1^T \dot{\underline{r}}_1 = \underline{\mathcal{F}}_2^T \dot{\underline{r}}_2 + \underline{\omega}_{21} \times \dot{\underline{r}} \\ &= \underline{\mathcal{F}}_2^T \dot{\underline{r}}_2 + \underline{\mathcal{F}}_2^T \underline{\omega}_2^{21 \times} \underline{r}_2 \\ &= \underline{\mathcal{F}}_2^T (\dot{\underline{r}}_2 + \underline{\omega}_2^{21 \times} \underline{r}_2). \end{aligned} \quad (6.41)$$

If we want to express the ‘inertial time derivative’ (that seen in $\underline{\mathcal{F}}_1$) in $\underline{\mathcal{F}}_1$, then we can use the rotation matrix \mathbf{C}_{12} :

$$\dot{\underline{r}}_1 = \mathbf{C}_{12} (\dot{\underline{r}}_2 + \underline{\omega}_2^{21 \times} \underline{r}_2). \quad (6.42)$$

Acceleration

Let us denote the *velocity* by

$$\underline{v} = \dot{\underline{r}}^\bullet = \dot{\underline{r}}^\circ + \underline{\omega}_{21} \times \dot{\underline{r}}.$$

The *acceleration* can be calculated by applying (6.39) to \underline{v} :

$$\begin{aligned}\underline{r}^{\bullet\bullet} &= \underline{v}^{\bullet} = \underline{v}^{\circ} + \underline{\omega}_{21} \times \underline{v} \\ &= (\underline{r}^{\circ\circ} + \underline{\omega}_{21} \times \underline{r}^{\circ} + \underline{\omega}_{21}^{\circ} \times \underline{r}) \\ &\quad + (\underline{\omega}_{21} \times \underline{r}^{\circ} + \underline{\omega}_{21} \times (\underline{\omega}_{21} \times \underline{r})) \\ &= \underline{r}^{\circ\circ} + 2\underline{\omega}_{21} \times \underline{r}^{\circ} + \underline{\omega}_{21}^{\circ} \times \underline{r} + \underline{\omega}_{21} \times (\underline{\omega}_{21} \times \underline{r}).\end{aligned}\tag{6.43}$$

The matrix equivalent in terms of components can be had by making the following substitutions:

$$\underline{r}^{\bullet\bullet} = \underline{\mathcal{F}}_1^T \ddot{\underline{r}}_1, \quad \underline{r}^{\circ\circ} = \underline{\mathcal{F}}_2^T \ddot{\underline{r}}_2, \quad \underline{\omega}_{21}^{\circ} = \underline{\mathcal{F}}_2^T \dot{\underline{\omega}}_2^{21}.$$

The result for the components is

$$\ddot{\underline{r}}_1 = \mathbf{C}_{12} \left[\ddot{\underline{r}}_2 + 2\omega_2^{21\times} \dot{\underline{r}}_2 + \dot{\underline{\omega}}_2^{21\times} \underline{r}_2 + \omega_2^{21\times} \omega_2^{21\times} \underline{r}_2 \right].\tag{6.44}$$

The various terms in the expression for the acceleration have been given special names:

- $\underline{r}^{\circ\circ}$: acceleration with respect to $\underline{\mathcal{F}}_2$
- $2\underline{\omega}_{21} \times \underline{r}^{\circ}$: Coriolis acceleration
- $\underline{\omega}_{21}^{\circ} \times \underline{r}$: angular acceleration
- $\underline{\omega}_{21} \times (\underline{\omega}_{21} \times \underline{r})$: centripetal acceleration

Angular Velocity Given Rotation Matrix

Begin with (6.3), which relates two reference frames via the rotation matrix:

$$\underline{\mathcal{F}}_1^T = \underline{\mathcal{F}}_2^T \mathbf{C}_{21}.$$

Now take the time derivative of both sides as seen in $\underline{\mathcal{F}}_1$:

$$\underline{0} = \underline{\mathcal{F}}_2^T \mathbf{C}_{21} + \underline{\mathcal{F}}_2^T \dot{\mathbf{C}}_{21}.$$

Substitute (6.36) for $\underline{\mathcal{F}}_2^T$:

$$\underline{0} = \underline{\omega}_{21} \times \underline{\mathcal{F}}_2^T \mathbf{C}_{21} + \underline{\mathcal{F}}_2^T \dot{\mathbf{C}}_{21}.$$

Now use (6.40) to get

$$\begin{aligned}\underline{0} &= \omega_2^{21^T} \underline{\mathcal{F}}_2 \times \underline{\mathcal{F}}_2^T \mathbf{C}_{21} + \underline{\mathcal{F}}_2^T \dot{\mathbf{C}}_{21} \\ &= \underline{\mathcal{F}}_2^T \left(\omega_2^{21\times} \mathbf{C}_{21} + \dot{\mathbf{C}}_{21} \right).\end{aligned}$$

Therefore, we conclude that

$$\dot{\mathbf{C}}_{21} = -\omega_2^{21\times} \mathbf{C}_{21},\tag{6.45}$$

which is known as *Poisson's equation*. Given the angular velocity as

Siméon Denis Poisson (1781-1840) was a French mathematician, geometer, and physicist.

measured in the frame $\underline{\mathcal{F}}_2$, the rotation matrix relating $\underline{\mathcal{F}}_1$ to $\underline{\mathcal{F}}_2$ can be determined by integrating the above expression⁴.

We can also rearrange to obtain an explicit function of ω_2^{21} :

$$\begin{aligned}\omega_2^{21\times} &= -\dot{\mathbf{C}}_{21}\mathbf{C}_{21}^{-1} \\ &= -\dot{\mathbf{C}}_{21}\mathbf{C}_{21}^T,\end{aligned}\quad (6.46)$$

which gives the angular velocity when the rotation matrix is known as a function of time.

Euler Angles

Consider the 1-2-3 Euler angle sequence and its associated rotation matrix. In this case, (6.46) becomes

$$\omega_2^{21\times} = -\mathbf{C}_3\mathbf{C}_2\dot{\mathbf{C}}_1\mathbf{C}_1^T\mathbf{C}_2^T\mathbf{C}_3^T - \mathbf{C}_3\dot{\mathbf{C}}_2\mathbf{C}_2^T\mathbf{C}_3^T - \dot{\mathbf{C}}_3\mathbf{C}_3^T. \quad (6.47)$$

Then, using

$$-\dot{\mathbf{C}}_i\mathbf{C}_i^T = \mathbf{1}_i^\times \dot{\theta}_i, \quad (6.48)$$

for each principal axis rotation (where $\mathbf{1}_i$ is column i of $\mathbf{1}$) and the identity

$$(\mathbf{C}_i\mathbf{r})^\times \equiv \mathbf{C}_i\mathbf{r}^\times\mathbf{C}_i^T, \quad (6.49)$$

we can show that

$$\omega_2^{21\times} = (\mathbf{C}_3\mathbf{C}_2\mathbf{1}_1\dot{\theta}_1)^\times + (\mathbf{C}_3\mathbf{1}_2\dot{\theta}_2)^\times + (\mathbf{1}_3\dot{\theta}_3)^\times, \quad (6.50)$$

which can be simplified to

$$\begin{aligned}\omega_2^{21} &= \underbrace{[\mathbf{C}_3(\theta_3)\mathbf{C}_2(\theta_2)\mathbf{1}_1 \quad \mathbf{C}_3(\theta_3)\mathbf{1}_2 \quad \mathbf{1}_3]}_{\mathbf{S}(\theta_2, \theta_3)} \underbrace{\begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix}}_{\dot{\boldsymbol{\theta}}} \\ &= \mathbf{S}(\theta_2, \theta_3)\dot{\boldsymbol{\theta}},\end{aligned}\quad (6.51)$$

which gives the angular velocity in terms of the Euler angles and the *Euler rates*, $\dot{\boldsymbol{\theta}}$. In scalar detail we have

$$\mathbf{S}(\theta_2, \theta_3) = \begin{bmatrix} \cos \theta_2 \cos \theta_3 & \sin \theta_3 & 0 \\ -\cos \theta_2 \sin \theta_3 & \cos \theta_3 & 0 \\ \sin \theta_2 & 0 & 1 \end{bmatrix}. \quad (6.52)$$

⁴ This is termed ‘strapdown navigation’ because the sensors that measure ω_2^{21} are strapped down in the rotating frame, $\underline{\mathcal{F}}_2$.

By inverting the matrix \mathbf{S} , we arrive at a system of differential equations that can be integrated to yield the Euler angles, assuming ω_2^{21} is known:

$$\begin{aligned}\dot{\theta} &= \mathbf{S}^{-1}(\theta_2, \theta_3)\omega_2^{21} \\ &= \begin{bmatrix} \sec \theta_2 \cos \theta_3 & -\sec \theta_2 \sin \theta_3 & 0 \\ \sin \theta_3 & \cos \theta_3 & 0 \\ -\tan \theta_2 \cos \theta_3 & \tan \theta_2 \sin \theta_3 & 1 \end{bmatrix} \omega_2^{21}. \quad (6.53)\end{aligned}$$

Note that \mathbf{S}^{-1} does not exist at $\theta_2 = \pi/2$, which is precisely the singularity associated with the 1-2-3 sequence.

It should be noted that the above developments hold true in general for any Euler sequence. If we pick an α - β - γ set,

$$\mathbf{C}_{21}(\theta_1, \theta_2, \theta_3) = \mathbf{C}_\gamma(\theta_3)\mathbf{C}_\beta(\theta_2)\mathbf{C}_\alpha(\theta_1), \quad (6.54)$$

then

$$\mathbf{S}(\theta_2, \theta_3) = [\mathbf{C}_\gamma(\theta_3)\mathbf{C}_\beta(\theta_2)\mathbf{1}_\alpha \quad \mathbf{C}_\gamma(\theta_3)\mathbf{1}_\beta \quad \mathbf{1}_\gamma], \quad (6.55)$$

and \mathbf{S}^{-1} does not exist at the singularities of \mathbf{S} .

6.2.5 Perturbing Rotations

Now that we have some basic notation built up for handling quantities in three-dimensional space, we will turn our focus to an issue that is often handled incorrectly or simply ignored altogether. We have shown in the previous section that the state of a single-body vehicle involves a translation, which has three degrees of freedom, as well as a rotation, which also has three degrees of freedom. The problem is that the degrees of freedom associated with rotations are a bit unique and must be handled carefully. The reason is that rotations do not live in a *vector space*⁵; rather, they form the *non-commutative group* called $SO(3)$.

As we have seen above, there are many ways of representing rotations mathematically, including rotation matrices, axis-angle formulations, Euler angles, and Euler parameters/unit-length quaternions. The most important fact to remember is that all these representations have the same underlying rotation, which only has three degrees of freedom. A 3×3 rotation matrix has nine elements, but only three are independent. Euler parameters have four scalar parameters, but only three are independent. Of all the common rotation representations, Euler angles are the only ones that have exactly three parameters; the problem is that Euler sequences have singularities, so for some problems, one must choose an appropriate sequence that avoids the singularities.

The fact that rotations do not live in a vector space is actually quite fundamental when it comes to linearizing motion and observation models involving rotations. What are we to do about linearizing rotations?

⁵ Here we mean a vector space in the sense of linear algebra.

Fortunately, there is a way forwards. The key is to consider what is happening on a small, in fact infinitesimal, level. We will begin by deriving a few key identities and then turn to linearizing a rotation matrix built from a sequence of Euler angles.

Some Key Identities

Euler's rotation theorem allows us to write a rotation matrix, \mathbf{C} , in terms of a rotation about an axis, \mathbf{a} , through an angle, ϕ :

$$\mathbf{C} = \cos \phi \mathbf{1} + (1 - \cos \phi) \mathbf{a} \mathbf{a}^T - \sin \phi \mathbf{a}^\times. \quad (6.56)$$

We now take the partial derivative of \mathbf{C} with respect to the angle, ϕ :

$$\frac{\partial \mathbf{C}}{\partial \phi} = -\sin \phi \mathbf{1} + \sin \phi \mathbf{a} \mathbf{a}^T - \cos \phi \mathbf{a}^\times \quad (6.57a)$$

$$= \sin \phi \underbrace{(-\mathbf{1} + \mathbf{a} \mathbf{a}^T)}_{\mathbf{a}^\times \mathbf{a}^\times} - \cos \phi \mathbf{a}^\times \quad (6.57b)$$

$$= -\cos \phi \mathbf{a}^\times - (1 - \cos \phi) \underbrace{\mathbf{a}^\times \mathbf{a} \mathbf{a}^T}_{\mathbf{0}} + \sin \phi \mathbf{a}^\times \mathbf{a}^\times \quad (6.57c)$$

$$= -\mathbf{a}^\times \underbrace{(\cos \phi \mathbf{1} + (1 - \cos \phi) \mathbf{a} \mathbf{a}^T - \sin \phi \mathbf{a}^\times)}_{\mathbf{C}}. \quad (6.57d)$$

Thus, our first important identity is

$$\frac{\partial \mathbf{C}}{\partial \phi} \equiv -\mathbf{a}^\times \mathbf{C}. \quad (6.58)$$

An immediate application of this is that for any principal-axis rotation, about axis α , we have

$$\frac{\partial \mathbf{C}_\alpha(\theta)}{\partial \theta} \equiv -\mathbf{1}_\alpha^\times \mathbf{C}_\alpha(\theta), \quad (6.59)$$

where $\mathbf{1}_\alpha$ is column α of the identity matrix.

Let us now consider an α - β - γ Euler sequence:

$$\mathbf{C}(\boldsymbol{\theta}) = \mathbf{C}_\gamma(\theta_3) \mathbf{C}_\beta(\theta_2) \mathbf{C}_\alpha(\theta_1), \quad (6.60)$$

where $\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3)$. Furthermore, we select an arbitrary constant vector, \mathbf{v} . Applying (6.59), we have

$$\frac{\partial (\mathbf{C}(\boldsymbol{\theta}) \mathbf{v})}{\partial \theta_3} = -\mathbf{1}_\gamma^\times \mathbf{C}_\gamma(\theta_3) \mathbf{C}_\beta(\theta_2) \mathbf{C}_\alpha(\theta_1) \mathbf{v} = (\mathbf{C}(\boldsymbol{\theta}) \mathbf{v})^\times \mathbf{1}_\gamma, \quad (6.61a)$$

$$\frac{\partial (\mathbf{C}(\boldsymbol{\theta}) \mathbf{v})}{\partial \theta_2} = -\mathbf{C}_\gamma(\theta_3) \mathbf{1}_\beta^\times \mathbf{C}_\beta(\theta_2) \mathbf{C}_\alpha(\theta_1) \mathbf{v} = (\mathbf{C}(\boldsymbol{\theta}) \mathbf{v})^\times \mathbf{C}_\gamma(\theta_3) \mathbf{1}_\beta, \quad (6.61b)$$

$$\frac{\partial (\mathbf{C}(\boldsymbol{\theta}) \mathbf{v})}{\partial \theta_1} = -\mathbf{C}_\gamma(\theta_3) \mathbf{C}_\beta(\theta_2) \mathbf{1}_\alpha^\times \mathbf{C}_\alpha(\theta_1) \mathbf{v} = (\mathbf{C}(\boldsymbol{\theta}) \mathbf{v})^\times \mathbf{C}_\gamma(\theta_3) \mathbf{C}_\beta(\theta_2) \mathbf{1}_\alpha, \quad (6.61c)$$

where we have made use of the two general identities

$$\mathbf{r}^\times \mathbf{s} \equiv -\mathbf{s}^\times \mathbf{r}, \quad (6.62a)$$

$$(\mathbf{R}\mathbf{s})^\times \equiv \mathbf{R}\mathbf{s}^\times \mathbf{R}^T \quad (6.62b)$$

for any vectors \mathbf{r} , \mathbf{s} and any rotation matrix \mathbf{R} . Combining the results in (6.61), we have

$$\begin{aligned} \frac{\partial (\mathbf{C}(\boldsymbol{\theta})\mathbf{v})}{\partial \boldsymbol{\theta}} &= \left[\frac{\partial(\mathbf{C}(\boldsymbol{\theta})\mathbf{v})}{\partial \theta_1} \quad \frac{\partial(\mathbf{C}(\boldsymbol{\theta})\mathbf{v})}{\partial \theta_2} \quad \frac{\partial(\mathbf{C}(\boldsymbol{\theta})\mathbf{v})}{\partial \theta_3} \right] \\ &= (\mathbf{C}(\boldsymbol{\theta})\mathbf{v})^\times \underbrace{[\mathbf{C}_\gamma(\theta_3)\mathbf{C}_\beta(\theta_2)\mathbf{1}_\alpha \quad \mathbf{C}_\gamma(\theta_3)\mathbf{1}_\beta \quad \mathbf{1}_\gamma]}_{\mathbf{S}(\theta_2, \theta_3)}, \end{aligned} \quad (6.63)$$

and thus another very important identity that we can state is

$$\frac{\partial (\mathbf{C}(\boldsymbol{\theta})\mathbf{v})}{\partial \boldsymbol{\theta}} \equiv (\mathbf{C}(\boldsymbol{\theta})\mathbf{v})^\times \mathbf{S}(\theta_2, \theta_3), \quad (6.64)$$

which we note is true regardless of the choice of Euler set. This will prove critical in the next section, when we discuss linearization of a rotation matrix.

Perturbing a Rotation Matrix

Let us return to first principles and consider carefully how to linearize a rotation. If we have a function, $\mathbf{f}(\mathbf{x})$, of some variable, \mathbf{x} , then perturbing \mathbf{x} slightly from its nominal value, $\bar{\mathbf{x}}$, by an amount $\delta\mathbf{x}$ will result in a change in the function. We can express this in terms of a Taylor-series expansion of \mathbf{f} about $\bar{\mathbf{x}}$:

$$\mathbf{f}(\bar{\mathbf{x}} + \delta\mathbf{x}) = \mathbf{f}(\bar{\mathbf{x}}) + \left. \frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} \right|_{\bar{\mathbf{x}}} \delta\mathbf{x} + (\text{higher order terms}) \quad (6.65)$$

and so if $\delta\mathbf{x}$ is small, a ‘first-order’ approximation is

$$\mathbf{f}(\bar{\mathbf{x}} + \delta\mathbf{x}) \approx \mathbf{f}(\bar{\mathbf{x}}) + \left. \frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} \right|_{\bar{\mathbf{x}}} \delta\mathbf{x}. \quad (6.66)$$

This presupposes that $\delta\mathbf{x}$ is not constrained in any way. The trouble with carrying out the same process with rotations is that most of the representations involve constraints and thus are not easily perturbed (without enforcing the constraint). The notable exceptions are the Euler angle sets. These contain exactly three parameters, and thus each can be varied independently. For this reason, we choose to use Euler angles in our perturbation of functions involving rotations.

Consider perturbing $\mathbf{C}(\boldsymbol{\theta})\mathbf{v}$ with respect to Euler angles $\boldsymbol{\theta}$, where \mathbf{v} is an arbitrary constant vector. Letting $\bar{\boldsymbol{\theta}} = (\bar{\theta}_1, \bar{\theta}_2, \bar{\theta}_3)$ and $\delta\boldsymbol{\theta} = (\delta\theta_1, \delta\theta_2, \delta\theta_3)$, then applying a first-order Taylor-series approximation,

we have

$$\begin{aligned}
\mathbf{C}(\bar{\boldsymbol{\theta}} + \delta\boldsymbol{\theta})\mathbf{v} &\approx \mathbf{C}(\bar{\boldsymbol{\theta}})\mathbf{v} + \frac{\partial (\mathbf{C}(\boldsymbol{\theta})\mathbf{v})}{\partial \boldsymbol{\theta}} \Big|_{\bar{\boldsymbol{\theta}}} \delta\boldsymbol{\theta} \\
&= \mathbf{C}(\bar{\boldsymbol{\theta}})\mathbf{v} + \left((\mathbf{C}(\boldsymbol{\theta})\mathbf{v})^\times \mathbf{S}(\theta_2, \theta_3) \right) \Big|_{\bar{\boldsymbol{\theta}}} \delta\boldsymbol{\theta} \\
&= \mathbf{C}(\bar{\boldsymbol{\theta}})\mathbf{v} + (\mathbf{C}(\bar{\boldsymbol{\theta}})\mathbf{v})^\times \mathbf{S}(\bar{\theta}_2, \bar{\theta}_3) \delta\boldsymbol{\theta} \\
&= \mathbf{C}(\bar{\boldsymbol{\theta}})\mathbf{v} - (\mathbf{S}(\bar{\theta}_2, \bar{\theta}_3) \delta\boldsymbol{\theta})^\times (\mathbf{C}(\bar{\boldsymbol{\theta}})\mathbf{v}) \\
&= \left(\mathbf{1} - (\mathbf{S}(\bar{\theta}_2, \bar{\theta}_3) \delta\boldsymbol{\theta})^\times \right) \mathbf{C}(\bar{\boldsymbol{\theta}})\mathbf{v}, \tag{6.67}
\end{aligned}$$

where we have used (6.64) to get to the second line. Observing that \mathbf{v} is arbitrary, we can drop it from both sides and write

$$\mathbf{C}(\bar{\boldsymbol{\theta}} + \delta\boldsymbol{\theta}) \approx \underbrace{\left(\mathbf{1} - (\mathbf{S}(\bar{\theta}_2, \bar{\theta}_3) \delta\boldsymbol{\theta})^\times \right)}_{\text{infinitesimal rot. mat.}} \mathbf{C}(\bar{\boldsymbol{\theta}}), \tag{6.68}$$

which we see is the product (not the sum) of an infinitesimal rotation matrix and the unperturbed rotation matrix, $\mathbf{C}(\bar{\boldsymbol{\theta}})$. Notationally, it is simpler to write

$$\mathbf{C}(\bar{\boldsymbol{\theta}} + \delta\boldsymbol{\theta}) \approx (\mathbf{1} - \delta\boldsymbol{\phi}^\times) \mathbf{C}(\bar{\boldsymbol{\theta}}), \tag{6.69}$$

with $\delta\boldsymbol{\phi} = \mathbf{S}(\bar{\theta}_2, \bar{\theta}_3) \delta\boldsymbol{\theta}$. Equation (6.68) is extremely important. It tells us exactly how to perturb a rotation matrix (in terms of perturbations to its Euler angles) when it appears inside any function.

Example 6.1 The following example shows how we can apply our linearized rotation expression in an arbitrary expression. Suppose we have a scalar function, J , given by

$$J(\boldsymbol{\theta}) = \mathbf{u}^T \mathbf{C}(\boldsymbol{\theta})\mathbf{v}, \tag{6.70}$$

where \mathbf{u} and \mathbf{v} are arbitrary vectors. Applying our approach to linearizing rotations, we have

$$J(\bar{\boldsymbol{\theta}} + \delta\boldsymbol{\theta}) \approx \mathbf{u}^T (\mathbf{1} - \delta\boldsymbol{\phi}^\times) \mathbf{C}(\bar{\boldsymbol{\theta}})\mathbf{v} = \underbrace{\mathbf{u}^T \mathbf{C}(\bar{\boldsymbol{\theta}})\mathbf{v}}_{J(\bar{\boldsymbol{\theta}})} + \underbrace{\mathbf{u}^T (\mathbf{C}(\bar{\boldsymbol{\theta}})\mathbf{v})^\times \delta\boldsymbol{\phi}}_{\delta J(\delta\boldsymbol{\theta})}, \tag{6.71}$$

so that the linearized function is

$$\delta J(\delta\boldsymbol{\theta}) = \underbrace{\left(\mathbf{u}^T (\mathbf{C}(\bar{\boldsymbol{\theta}})\mathbf{v})^\times \mathbf{S}(\bar{\theta}_2, \bar{\theta}_3) \right)}_{\text{constant}} \delta\boldsymbol{\theta}, \tag{6.72}$$

where we see that the factor in front of $\delta\boldsymbol{\theta}$ is indeed constant; in fact, it is $\frac{\partial J}{\partial \boldsymbol{\theta}}|_{\bar{\boldsymbol{\theta}}}$, the Jacobian of J with respect to $\boldsymbol{\theta}$.

6.3 Poses

We have spent considerable effort discussing the *rotational* aspect of a moving body. We now introduce the notation of *translation*. Together, the translation and rotation of a body are referred to as the *pose*. Pose estimation problems are often concerned with transforming the coordinates of a point, P , between a moving (in translation and rotation) vehicle frame, and a stationary frame, as depicted in Figure 6.2.

We can relate the vectors in Figure 6.2 as follows:

$$\overrightarrow{r}_i^{pi} = \overrightarrow{r}_i^{pv} + \overrightarrow{r}_i^{vi}, \quad (6.73)$$

where we have not yet selected any particular reference frame in which to express the relationship. Writing the relationship in the stationary frame, $\underline{\mathcal{F}}_i$, we have

$$\mathbf{r}_i^{pi} = \mathbf{r}_i^{pv} + \mathbf{r}_i^{vi}. \quad (6.74)$$

If the point, P , is attached to the vehicle, we typically know its coordinates in $\underline{\mathcal{F}}_v$, which is rotated with respect to $\underline{\mathcal{F}}_i$. Letting \mathbf{C}_{iv} represent this rotation, we can rewrite the relationship as

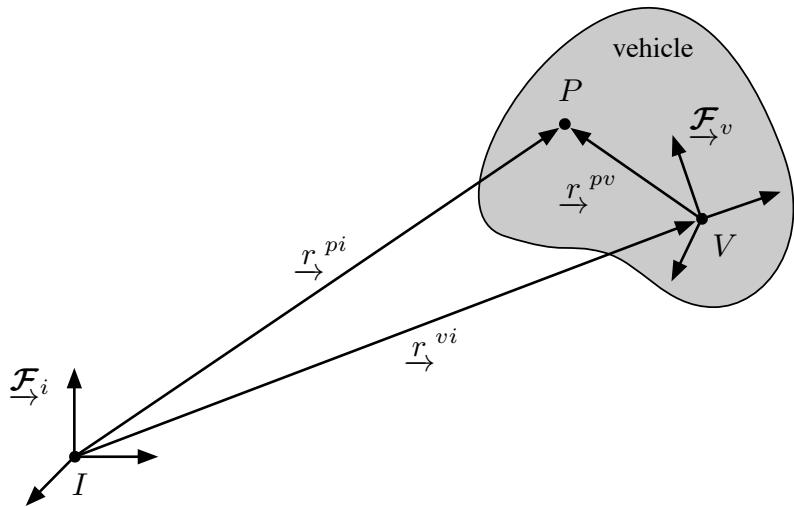
$$\mathbf{r}_i^{pi} = \mathbf{C}_{iv} \mathbf{r}_v^{pv} + \mathbf{r}_i^{vi}, \quad (6.75)$$

which tells us how to convert the coordinates of P in $\underline{\mathcal{F}}_v$ to its coordinates in $\underline{\mathcal{F}}_i$, given knowledge of the translation, \mathbf{r}_i^{vi} , and rotation, \mathbf{C}_{iv} , between the two frames. We will refer to

$$\{\mathbf{r}_i^{vi}, \mathbf{C}_{iv}\}, \quad (6.76)$$

as the *pose* of the vehicle.

Figure 6.2 Pose estimation problems are often concerned with transforming the coordinates of a point, P , between a moving vehicle frame, and a stationary frame.



6.3.1 Transformation Matrices

We can also write the relationship expressed in (6.75) in another convenient form:

$$\begin{bmatrix} \mathbf{r}_i^{pi} \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{C}_{iv} & \mathbf{r}_i^{vi} \\ \mathbf{0}^T & 1 \end{bmatrix}}_{\mathbf{T}_{iv}} \begin{bmatrix} \mathbf{r}_v^{pv} \\ 1 \end{bmatrix}, \quad (6.77)$$

where \mathbf{T}_{iv} is referred to as a 4×4 *transformation matrix*.

To make use of a transformation matrix, we must augment the coordinates of a point with a 1,

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \quad (6.78)$$

which is referred to as a *homogeneous* point representation. An interesting property of homogeneous point representations is that each entry can be multiplied by a *scale factor*, s :

$$\begin{bmatrix} sx \\ sy \\ sz \\ s \end{bmatrix}. \quad (6.79)$$

To recover the original (x, y, z) coordinates, one needs only to divide the first three entries by the fourth. In this way, as the scale factor approaches 0, we can represent points arbitrarily far away from the origin. Hartley and Zisserman (2000) discuss the use of homogeneous coordinates at length for computer-vision applications.

To transform the coordinates back the other way, we require the inverse of a transformation matrix:

$$\begin{bmatrix} \mathbf{r}_v^{pv} \\ 1 \end{bmatrix} = \mathbf{T}_{iv}^{-1} \begin{bmatrix} \mathbf{r}_i^{pi} \\ 1 \end{bmatrix}, \quad (6.80)$$

where

$$\begin{aligned} \mathbf{T}_{iv}^{-1} &= \begin{bmatrix} \mathbf{C}_{iv} & \mathbf{r}_i^{vi} \\ \mathbf{0}^T & 1 \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{C}_{iv}^T & -\mathbf{C}_{iv}^T \mathbf{r}_i^{vi} \\ \mathbf{0}^T & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{C}_{vi} & -\mathbf{r}_v^{vi} \\ \mathbf{0}^T & 1 \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{C}_{vi} & \mathbf{r}_v^{iv} \\ \mathbf{0}^T & 1 \end{bmatrix} = \mathbf{T}_{vi}, \end{aligned} \quad (6.81)$$

where we have used that $\mathbf{r}_v^{iv} = -\mathbf{r}_v^{vi}$, which simply flips the direction of the vector.

We can also compound transformation matrices:

$$\mathbf{T}_{iv} = \mathbf{T}_{ia} \mathbf{T}_{ab} \mathbf{T}_{bv}, \quad (6.82)$$

Homogeneous coordinates were introduced by Augustus Ferdinand Möbius (1790-1868) in his work entitled *Der Barycentrische Calcul*, published in 1827. Möbius parameterized a point on a plane, (x, y) , by considering masses, m_1 , m_2 , and m_3 , that must be placed at the vertices of a fixed triangle to make the point the triangle's center of mass. The coordinates (m_1, m_2, m_3) are not unique, as scaling the three masses equally does not change the point location. When the equation of a curve is written in this coordinate system, it becomes homogeneous in (m_1, m_2, m_3) . For example, a circle centered at (a, b) with radius r is: $(x-a)^2 + (y-b)^2 = r^2$. Written in homogeneous coordinates with $x = m_1/m_3$ and $y = m_2/m_3$, the equation becomes $(m_1 - m_3a)^2 + (m_2 - m_3b)^2 = m_3^2 r^2$, where every term is now quadratic in the homogeneous coordinates (Furgale, 2011).

which makes it easy to chain an arbitrary number of pose changes together:

$$\underline{\mathcal{F}}_i \xleftarrow{\mathbf{T}_{iv}} \underline{\mathcal{F}}_v = \underline{\mathcal{F}}_i \xleftarrow{\mathbf{T}_{ia}} \underline{\mathcal{F}}_a \xleftarrow{\mathbf{T}_{ab}} \underline{\mathcal{F}}_b \xleftarrow{\mathbf{T}_{bv}} \underline{\mathcal{F}}_v \quad (6.83)$$

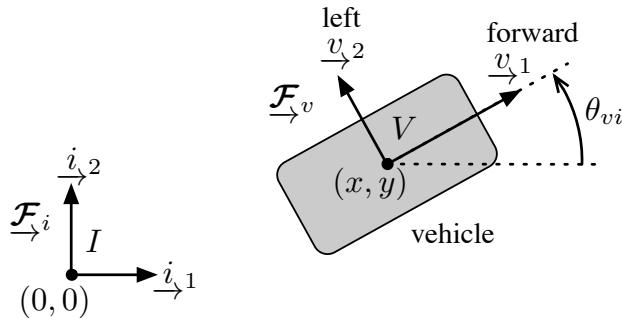
For example, each frame could represent the pose of a mobile vehicle at a different instant in time, and this relation tells us how to combine relative motions into a global one.

Transformation matrices are very appealing because they tell us to first apply the translation and then the rotation. This is often a source of ambiguity when working with poses because the subscripts and superscripts are typically dropped in practice, and then it is difficult to know the exact meaning of each quantity.

6.3.2 Robotics Conventions

There is an important subtlety that must be mentioned to conform with standard practice in robotics. We can understand this in the context of a simple example. Imagine a vehicle travelling in the xy -plane, as depicted in Figure 6.3.

Figure 6.3
Simple planar example with a mobile vehicle whose state is given by position, (x, y) , and orientation, θ_{vi} . It is standard for ‘forward’ to be the 1-axis of the vehicle frame and ‘left’ to be the 2-axis. Note that the 3-axis is coming out of the page.



The position of the vehicle can be written in a straightforward manner as

$$\mathbf{r}_i^{vi} = \begin{bmatrix} x \\ y \\ 0 \end{bmatrix}. \quad (6.84)$$

The z -coordinate is zero for this planar example.

The rotation of $\underline{\mathcal{F}}_v$ with respect to $\underline{\mathcal{F}}_i$ is a principal-axis rotation about the 3-axis, through an angle θ_{vi} (we add the subscript to demonstrate a point). Following our convention from before, the angle of rotation is positive (according to the right-hand rule). Thus, we have

$$\mathbf{C}_{vi} = \mathbf{C}_3(\theta_{vi}) = \begin{bmatrix} \cos \theta_{vi} & \sin \theta_{vi} & 0 \\ -\sin \theta_{vi} & \cos \theta_{vi} & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (6.85)$$

It makes sense to use θ_{vi} for orientation; it naturally describes the heading of the vehicle since it is $\underline{\mathcal{F}}_v$ that is moving with respect to $\underline{\mathcal{F}}_i$. However, as discussed in the last section, the rotation matrix that we really care about when constructing the pose is $\mathbf{C}_{iv} = \mathbf{C}_{vi}^T = \mathbf{C}_3(-\theta_{vi}) = \mathbf{C}_3(\theta_{iv})$. Importantly, we note that $\theta_{iv} = -\theta_{vi}$. We do not want to use θ_{iv} as the heading as that will be quite confusing.

Sticking with θ_{vi} , the pose of the vehicle can then be written in transformation matrix form as

$$\mathbf{T}_{iv} = \begin{bmatrix} \mathbf{C}_{iv} & \mathbf{r}_i^{vi} \\ \mathbf{0}^T & 1 \end{bmatrix} = \begin{bmatrix} \cos \theta_{vi} & -\sin \theta_{vi} & 0 & x \\ \sin \theta_{vi} & \cos \theta_{vi} & 0 & y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (6.86)$$

which is perfectly fine. In general, even when the axis of rotation, \mathbf{a} , is not \underline{i}_3 , we are free to write

$$\begin{aligned} \mathbf{C}_{iv} = \mathbf{C}_{vi}^T &= (\cos \theta_{vi} \mathbf{1} + (1 - \cos \theta_{vi}) \mathbf{a} \mathbf{a}^T - \sin \theta_{vi} \mathbf{a}^\times)^T \\ &= \cos \theta_{vi} \mathbf{1} + (1 - \cos \theta_{vi}) \mathbf{a} \mathbf{a}^T + \sin \theta_{vi} \mathbf{a}^\times, \end{aligned} \quad (6.87)$$

where we note the change in sign of the third term due to the skew-symmetric property, $\mathbf{a}^{\times^T} = -\mathbf{a}^\times$. In other words, we are free to use θ_{vi} rather than θ_{iv} to construct \mathbf{C}_{iv} .

Confusion arises, however, when all the subscripts are dropped and we simply write

$$\mathbf{C} = \cos \theta \mathbf{1} + (1 - \cos \theta) \mathbf{a} \mathbf{a}^T + \sin \theta \mathbf{a}^\times, \quad (6.88)$$

which is very common in robotics. There is absolutely nothing wrong with this expression; we must simply realize that when written in this form, the rotation is the other way around from our earlier development⁶.

There is another slight change in notation that is common in robotics as well. Often, the $(\cdot)^\times$ symbol is replaced with the $(\cdot)^\wedge$ symbol (Murray et al., 1994), particularly when dealing with transformation matrices. The expression for a rotation matrix is then written as

$$\mathbf{C} = \cos \theta \mathbf{1} + (1 - \cos \theta) \mathbf{a} \mathbf{a}^T + \sin \theta \mathbf{a}^\wedge. \quad (6.89)$$

We need to be quite careful with angular velocity as well, since this should in some way match the convention we are using for the angle of rotation.

⁶ Our goal in this section is to make things clear, rather than to argue in favour of one convention over another. However, it is worth noting that this convention, with the third term in (6.88) positive, is conforming to a left-hand rotation rather than a right-hand one.

Finally, the pose is written as

$$\mathbf{T} = \begin{bmatrix} \mathbf{C} & \mathbf{r} \\ \mathbf{0}^T & 1 \end{bmatrix}, \quad (6.90)$$

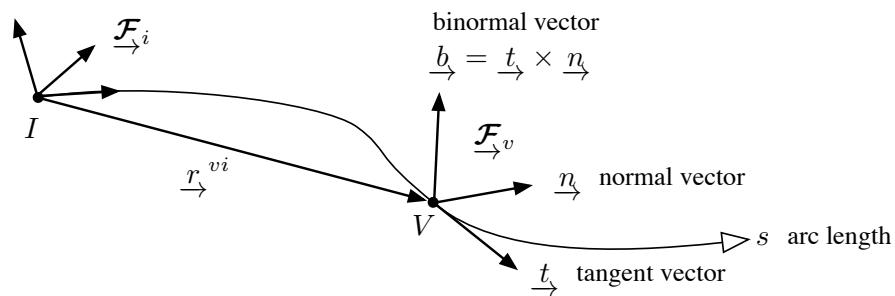
with all the subscripts removed. We simply need to be careful to remember what all of the quantities actually mean when using them in practice.

In an effort to be relevant to robotics, we will adopt the conventions in (6.89) moving forward in this book. However, we believe it has been worthwhile to begin at first principles to better understand what all the quantities associated with pose really mean.

6.3.3 Frenet-Serret Frame

It is worth drawing the connection between our pose variables (represented as transformation matrices) and the classical *Frenet-Serret* moving frame. Figure 6.4 depicts a point, V , moving smoothly through space. The Frenet-Serret frame is attached to the point with the first axis in the direction of motion (the curve tangent), the second axis pointing in the direction of the tangent derivative with respect to arc length (the curve normal), and the third axis completing the frame (the binormal).

Figure 6.4 The classical Frenet-Serret moving frame can be used to describe the motion of a point. The frame axes point in the tangent, normal, and binormal directions of the curve traced out by the point. This frame and its motion equations are named after the two French mathematicians who independently discovered them: Jean Frédéric Frenet, in his thesis of 1847, and Joseph Alfred Serret in 1851.



The Frenet-Serret equations describe how the frame axes change with arc length:

$$\frac{d}{ds} \vec{t} = \kappa \vec{n}, \quad (6.91a)$$

$$\frac{d}{ds} \vec{n} = -\kappa \vec{t} + \tau \vec{b}, \quad (6.91b)$$

$$\frac{d}{ds} \vec{b} = -\tau \vec{n}, \quad (6.91c)$$

where κ is called the *curvature* of the path and τ is called the *torsion*

of the path. Stacking the axes into a frame, $\underline{\mathcal{F}}_v$,

$$\underline{\mathcal{F}}_v = \begin{bmatrix} \underline{t} \\ \underline{n} \\ \underline{b} \end{bmatrix}, \quad (6.92)$$

we can write the Frenet-Serret equations as

$$\frac{d}{ds} \underline{\mathcal{F}}_v = \begin{bmatrix} 0 & \kappa & 0 \\ -\kappa & 0 & \tau \\ 0 & -\tau & 0 \end{bmatrix} \underline{\mathcal{F}}_v. \quad (6.93)$$

Multiplying both sides by the speed along the path, $v = ds/dt$, and right-multiplying by $\underline{\mathcal{F}}_i^T$, we have

$$\underbrace{\frac{d}{dt} (\underline{\mathcal{F}}_v \cdot \underline{\mathcal{F}}_i^T)}_{\dot{\mathbf{C}}_{vi}} = \underbrace{\begin{bmatrix} 0 & v\kappa & 0 \\ -v\kappa & 0 & v\tau \\ 0 & -v\tau & 0 \end{bmatrix}}_{-\boldsymbol{\omega}_v^{vi}\wedge} \underbrace{(\underline{\mathcal{F}}_v \cdot \underline{\mathcal{F}}_i^T)}_{\mathbf{C}_{vi}}, \quad (6.94)$$

where we have applied the chain rule. We see that this has recovered Poisson's equation for rotational kinematics as given previously in (6.45); the angular velocity expressed in the moving frame,

$$\boldsymbol{\omega}_v^{vi} = \begin{bmatrix} v\tau \\ 0 \\ v\kappa \end{bmatrix}, \quad (6.95)$$

is constrained to only two degrees of freedom since the middle entry is zero. We also have the translational kinematics,

$$\dot{\mathbf{r}}_i^{vi} = \mathbf{C}_{vi}^T \boldsymbol{\nu}_v^{vi}, \quad \boldsymbol{\nu}_v^{vi} = \begin{bmatrix} v \\ 0 \\ 0 \end{bmatrix}. \quad (6.96)$$

To express this in the body frame, we note that

$$\begin{aligned} \dot{\mathbf{r}}_v^{iv} &= \frac{d}{dt} (-\mathbf{C}_{vi} \mathbf{r}_i^{vi}) = -\dot{\mathbf{C}}_{vi} \mathbf{r}_i^{vi} - \mathbf{C}_{vi} \dot{\mathbf{r}}_i^{vi} \\ &= \boldsymbol{\omega}_v^{vi}\wedge \mathbf{C}_{vi} \mathbf{r}_i^{vi} - \mathbf{C}_{vi} \mathbf{C}_{vi}^T \boldsymbol{\nu}_v^{vi} = -\boldsymbol{\omega}_v^{vi}\wedge \mathbf{r}_v^{iv} - \boldsymbol{\nu}_v^{vi}. \end{aligned} \quad (6.97)$$

We can then combine the translational and rotational kinematics into transformation-matrix form as follows:

$$\begin{aligned} \dot{\mathbf{T}}_{vi} &= \frac{d}{dt} \begin{bmatrix} \mathbf{C}_{vi} & \mathbf{r}_v^{iv} \\ \mathbf{0}^T & 1 \end{bmatrix} = \begin{bmatrix} \dot{\mathbf{C}}_{vi} & \dot{\mathbf{r}}_v^{iv} \\ \mathbf{0}^T & 0 \end{bmatrix} = \begin{bmatrix} -\boldsymbol{\omega}_v^{vi}\wedge \mathbf{C}_{vi} & -\boldsymbol{\omega}_v^{vi}\wedge \mathbf{r}_v^{iv} - \boldsymbol{\nu}_v^{vi} \\ \mathbf{0}^T & 0 \end{bmatrix} \\ &= \begin{bmatrix} -\boldsymbol{\omega}_v^{vi}\wedge & -\boldsymbol{\nu}_v^{vi} \\ \mathbf{0}^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{C}_{vi} & \mathbf{r}_v^{iv} \\ \mathbf{0}^T & 1 \end{bmatrix} = \begin{bmatrix} 0 & v\kappa & 0 & -v \\ -v\kappa & 0 & v\tau & 0 \\ 0 & -v\tau & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \mathbf{T}_{vi}. \end{aligned} \quad (6.98)$$

Integrating this forward in time provides both the translation and rotation of the moving frame. We can think of (v, κ, τ) as three inputs in this case as they determine the shape of the curve that is traced out. We will see in the next chapter that these kinematic equations can be generalized to the form

$$\dot{\mathbf{T}} = \begin{bmatrix} \boldsymbol{\omega}^\wedge & \boldsymbol{\nu} \\ \mathbf{0}^T & 0 \end{bmatrix} \mathbf{T}, \quad (6.99)$$

where

$$\boldsymbol{\varpi} = \begin{bmatrix} \boldsymbol{\nu} \\ \boldsymbol{\omega} \end{bmatrix} \quad (6.100)$$

is a (slightly differently defined) generalized six-degree-of-freedom velocity vector (expressed in the moving frame) that allows for all possible curves for \mathbf{T} to be traced out. The Frenet-Serret equations can be viewed as a special case of this general kinematic formula.

If we want to use \mathbf{T}_{iv} (for reasons described in the last section) instead of \mathbf{T}_{vi} , we can either integrate the above and then output $\mathbf{T}_{iv} = \mathbf{T}_{vi}^{-1}$, or we can instead integrate

$$\dot{\mathbf{T}}_{iv} = \mathbf{T}_{iv} \begin{bmatrix} 0 & -v\kappa & 0 & v \\ v\kappa & 0 & -v\tau & 0 \\ 0 & v\tau & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad (6.101)$$

which will achieve the same result (proof left as an exercise). If we constrain the motion to the xy -plane, which can be achieved by setting the initial condition to

$$\mathbf{T}_{iv}(0) = \begin{bmatrix} \cos \theta_{vi}(0) & -\sin \theta_{vi}(0) & 0 & x(0) \\ \sin \theta_{vi}(0) & \cos \theta_{vi}(0) & 0 & y(0) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (6.102)$$

and then forcing $\tau = 0$ for all time, the kinematics simplify to

$$\dot{x} = v \cos \theta, \quad (6.103a)$$

$$\dot{y} = v \sin \theta, \quad (6.103b)$$

$$\dot{\theta} = \omega, \quad (6.103c)$$

where $\omega = v\kappa$ and it is understood that $\theta = \theta_{vi}$. This last model is sometimes referred to as the ‘unicycle model’ for a differential-drive mobile robot. The inputs are the longitudinal speed, v , and the rotational speed, ω . The robot is unable to translate sideways due to the nonholonomic constraint associated with its wheels; it can only roll forwards and turn.

6.4 Sensor Models

Now that we have some three-dimensional tools, we will introduce a few three-dimensional sensor models that can be used inside our state estimation algorithms. In general, we will be interested in sensors that are on-board our robot. This situation is depicted in Figure 6.5.

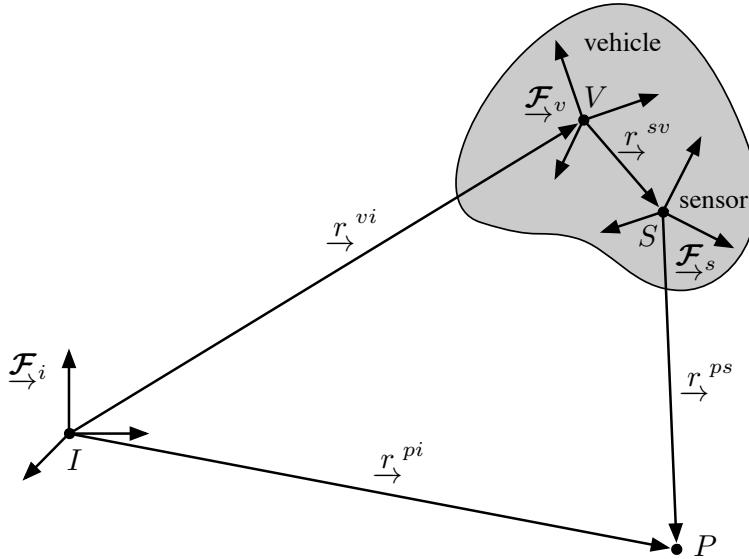


Figure 6.5
Reference frames for a moving vehicle with a sensor on-board that observes a point, P , in the world.

We have an inertial frame, \mathcal{F}_i , a vehicle frame, \mathcal{F}_v , and a sensor frame, \mathcal{F}_s . The pose change between the sensor frame and the vehicle frame, \mathbf{T}_{sv} , called the *extrinsic sensor parameters*, is typically fixed and is either determined through some form of separate calibration method or is folded directly into the state estimation procedure. In the sensor-model developments to follow, we will focus solely on how a point, P , is observed by a sensor attached to \mathcal{F}_s .

6.4.1 Perspective Camera

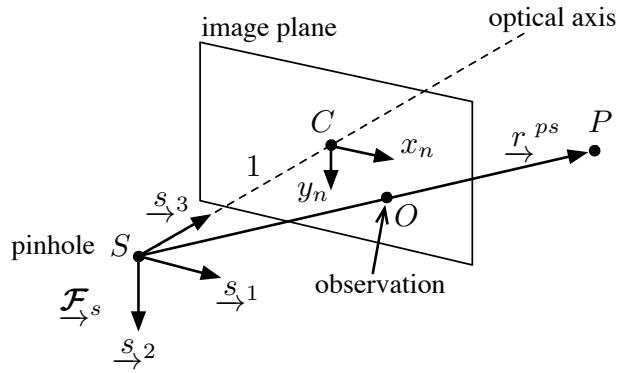
One of the most important sensors is the perspective camera as it is cheap yet can be used to infer motion of a vehicle and also the shape of the world.

Normalized Image Coordinates

Figure 6.6 depicts the observation, O , of a point, P , in an ideal perspective camera. In reality, the image plane is behind the pinhole, but showing it in front avoids the mental effort of working with a flipped image. This is called the *frontal projection model*. The s_3 axis of \mathcal{F}_s , called the *optical axis*, is orthogonal to the image plane and the dis-

Figure 6.6

Frontal projection model of a camera showing the observation, O , of a point, P , in the image plane. In reality, the image plane is behind the pinhole but the frontal model avoids flipping the image.



tance between the pinhole, S , and the image plane center, C , called the *focal length*, is 1 for this idealized camera model.

If the coordinates of P in \mathcal{F}_s are

$$\boldsymbol{\rho} = \mathbf{r}_s^{ps} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \quad (6.104)$$

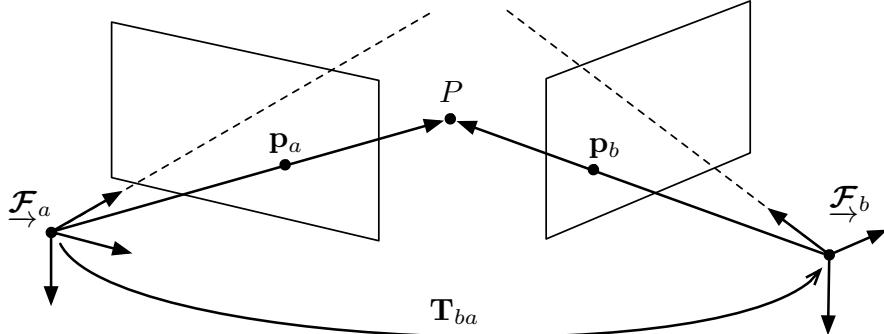
with the s_3 axis orthogonal to the image plane, then the coordinates of O in the image plane are

$$x_n = x/z, \quad (6.105a)$$

$$y_n = y/z. \quad (6.105b)$$

These are called the (two-dimensional) *normalized image coordinates*, and are sometimes provided in a homogeneous form as

$$\mathbf{p} = \begin{bmatrix} x_n \\ y_n \\ 1 \end{bmatrix}. \quad (6.106)$$

Figure 6.7 Two camera observations of the same point, P .

Essential Matrix

If a point, P , is observed by a camera, the camera moved, and then the same point observed again, the two normalized image coordinates corresponding to the observations, \mathbf{p}_a and \mathbf{p}_b , are related to one another through the following constraint:

$$\mathbf{p}_a^T \mathbf{E}_{ab} \mathbf{p}_b = 0, \quad (6.107)$$

where \mathbf{E}_{ab} is called the *essential matrix* (of computer vision),

$$\mathbf{E}_{ab} = \mathbf{C}_{ba}^T \mathbf{r}_b^{ab\wedge} \quad (6.108)$$

and is related to the pose change of the camera,

$$\mathbf{T}_{ba} = \begin{bmatrix} \mathbf{C}_{ba} & \mathbf{r}_b^{ab\wedge} \\ \mathbf{0}^T & 1 \end{bmatrix}. \quad (6.109)$$

To see that the constraint is true, we let

$$\mathbf{p}_j = \frac{1}{z_j} \boldsymbol{\rho}_j, \quad \boldsymbol{\rho}_j = \begin{bmatrix} x_j \\ y_j \\ z_j \end{bmatrix} \quad (6.110)$$

for $j = a, b$. We also have

$$\boldsymbol{\rho}_a = \mathbf{C}_{ba}^T (\boldsymbol{\rho}_b - \mathbf{r}_b^{ab}) \quad (6.111)$$

for the change in coordinates of P due to the camera moving. Then, returning to the constraint, we see

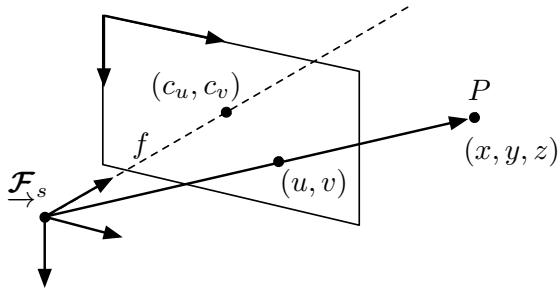
$$\begin{aligned} \mathbf{p}_a^T \mathbf{E}_{ab} \mathbf{p}_b &= \frac{1}{z_a z_b} \boldsymbol{\rho}_a^T \mathbf{E}_{ab} \boldsymbol{\rho}_b = \frac{1}{z_a z_b} (\boldsymbol{\rho}_b - \mathbf{r}_b^{ab})^T \underbrace{\mathbf{C}_{ba} \mathbf{C}_{ba}^T}_{1} \mathbf{r}_b^{ab\wedge} \boldsymbol{\rho}_b \\ &= \frac{1}{z_a z_b} \left(-\underbrace{\boldsymbol{\rho}_b^T \boldsymbol{\rho}_b^\wedge}_{0} \mathbf{r}_b^{ab} - \underbrace{\mathbf{r}_b^{abT} \mathbf{r}_b^{ab\wedge}}_{0} \boldsymbol{\rho}_b \right) = 0. \quad (6.112) \end{aligned}$$

The essential matrix can be useful in some pose-estimation problems, including camera calibration.

Lens Distortion

In general, lens effects can distort camera images so that the normalized image coordinate equations are only approximately true. A variety of analytical models of this distortion are available, and these can be used to correct the raw images such that the resulting images appear as though they come from an idealized pinhole camera, and thus the normalized image coordinate equations hold. We will assume this undistortion procedure has been applied to the images and avoid elaborating on the distortion models.

Figure 6.8
Camera model showing intrinsic parameters: f is the focal length, (c_u, c_v) is the optical axis intersection.



Intrinsic Parameters

The normalized image coordinates are really associated with a hypothetical camera with unit focal length and image origin at the optical axis intersection. We can map the normalized image coordinates, (x_n, y_n) , to the actual *pixel coordinates*, (u, v) , through the following relation:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} f_u & 0 & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{K}} \begin{bmatrix} x_n \\ y_n \\ 1 \end{bmatrix}, \quad (6.113)$$

where \mathbf{K} is called the *intrinsic parameter matrix* and contains the actual camera focal length expressed in horizontal pixels, f_u , and vertical pixels, f_v , as well as the actual offset of the image origin from the optical axis intersection, (c_u, c_v) , also expressed in horizontal, vertical pixels⁷. These intrinsic parameters are typically determined during the calibration procedure used to remove the lens effects, so that we can assume \mathbf{K} is known.

Fundamental Matrix

Similarly to the essential matrix constraint, there is a constraint that can be expressed between the homogeneous pixel coordinates of two observations of a point from different camera perspectives (and possibly even different cameras). Let

$$\mathbf{q}_i = \mathbf{K}_i \mathbf{p}_i, \quad (6.114)$$

with $i = a, b$ for the pixel coordinates of two camera observations with different intrinsic parameter matrices. Then the following constraint holds:

$$\mathbf{q}_a^T \mathbf{F}_{ab} \mathbf{q}_b = 0, \quad (6.115)$$

where

$$\mathbf{F}_{ab} = \mathbf{K}_a^{-T} \mathbf{E}_{ab} \mathbf{K}_b^{-1} \quad (6.116)$$

⁷ On many imaging sensors, the pixels are not square, resulting in different units in the horizontal and vertical directions.

is called the *fundamental matrix* (of computer vision). It is fairly easy to see the constraint is true by substitution:

$$\mathbf{q}_a^T \mathbf{F}_{ab} \mathbf{q}_b = \mathbf{p}_b^T \underbrace{\mathbf{K}_a^T \mathbf{K}_a^{-T}}_1 \mathbf{E}_{ab} \underbrace{\mathbf{K}_b^{-1} \mathbf{K}_b}_1 \mathbf{p}_b = 0, \quad (6.117)$$

where we use the essential-matrix constraint for the last step.

The constraint associated with the fundamental matrix is also sometimes called the *epipolar constraint* and is depicted geometrically in Figure 6.9. If a point is observed in one camera, \mathbf{q}_a , and the fundamental matrix between the first and a second camera is known, the constraint describes a line, called the *epipolar line*, along which the observation of the point in the second camera, \mathbf{q}_b , must lie. This property can be used to limit the search for a matching point to just the epipolar line. This is possible because the camera model is an *affine transformation*, implying that a straight line in Euclidean space projects to a straight line in image space. The fundamental matrix is also useful in developing methods to determine the intrinsic parameter matrix, for example.

Complete Model

Combining everything but the lens effects, the perspective camera model can be written as

$$\begin{bmatrix} u \\ v \end{bmatrix} = \mathbf{s}(\boldsymbol{\rho}) = \mathbf{P} \mathbf{K} \frac{1}{z} \boldsymbol{\rho}, \quad (6.118)$$

where

$$\mathbf{P} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \quad \mathbf{K} = \begin{bmatrix} f_u & 0 & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 1 \end{bmatrix}, \quad \boldsymbol{\rho} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}. \quad (6.119)$$

\mathbf{P} is simply a projection matrix to remove the bottom row from the homogeneous point representation. This form of the model makes it clear that with a single camera, there is a loss of information as we are

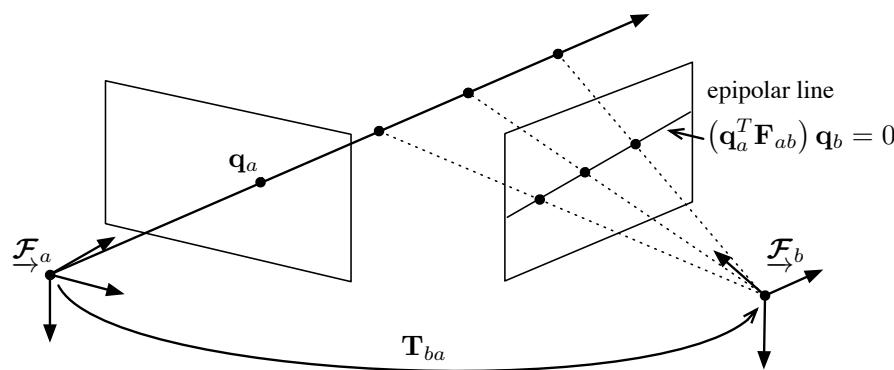


Figure 6.9 If a point is observed in one image, \mathbf{q}_a , and the fundamental matrix, \mathbf{F}_{ab} , is known, this can be used to define a line in the second image along which the second observation, \mathbf{q}_b , must lie.

going from three parameters in ρ to just two in (u, v) ; we are unable to determine depth from just one camera.

Homography

Although we cannot determine depth from just one camera, if we assume that the point a camera is observing lies on the surface of a plane whose geometry is known, we can work out the depth and then how that point will look to another camera. The geometry of this situation is depicted in Figure 6.10.

The homogeneous observations for the two cameras can be written as

$$\mathbf{q}_i = \mathbf{K}_i \frac{1}{z_i} \boldsymbol{\rho}_i, \quad \boldsymbol{\rho}_i = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix}, \quad (6.120)$$

where $\boldsymbol{\rho}_i$ are the coordinates of P in each camera frame with $i = a, b$. Let us assume we know the equation of the plane containing P , expressed in both camera frames; this can be parameterized as

$$\{\mathbf{n}_i, d_i\}, \quad (6.121)$$

where d_i is the distance of camera i from the plane and \mathbf{n}_i are the coordinates of the plane normal in frame i . This implies that

$$\mathbf{n}_i^T \boldsymbol{\rho}_i + d_i = 0, \quad (6.122)$$

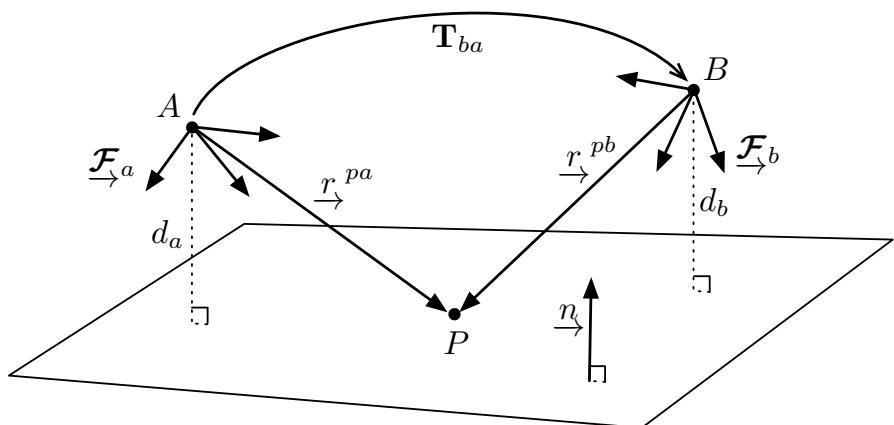
since P is in the plane. Solving for $\boldsymbol{\rho}_i$ in (6.120) and substituting into the plane equation, we have

$$z_i \mathbf{n}_i^T \mathbf{K}_i^{-1} \mathbf{q}_i + d_i = 0, \quad (6.123)$$

or

$$z_i = -\frac{d_i}{\mathbf{n}_i^T \mathbf{K}_i^{-1} \mathbf{q}_i} \quad (6.124)$$

Figure 6.10 If the point observed by a camera lies on a plane whose geometry is known, it is possible to work out what that point will look like after the camera makes a pose change using a transform called a homography.



for the depth of point P in each camera, $i = a, b$. This further implies that we can write the coordinates of P , expressed in each camera frame, as

$$\boldsymbol{\rho}_i = -\frac{d_i}{\mathbf{n}_i^T \mathbf{K}_i^{-1} \mathbf{q}_i} \mathbf{K}_i^{-1} \mathbf{q}_i. \quad (6.125)$$

This shows that the knowledge of the plane parameters, $\{\mathbf{n}_i, d_i\}$, allows us to recover the coordinates of P even though a single camera cannot determine depth on its own.

Let us also assume we know the pose change, \mathbf{T}_{ba} , from \mathcal{F}_a to \mathcal{F}_b so that

$$\begin{bmatrix} \boldsymbol{\rho}_b \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{C}_{ba} & \mathbf{r}_b^{ab} \\ \mathbf{0}^T & 1 \end{bmatrix}}_{\mathbf{T}_{ba}} \begin{bmatrix} \boldsymbol{\rho}_a \\ 1 \end{bmatrix}, \quad (6.126)$$

or

$$\boldsymbol{\rho}_b = \mathbf{C}_{ba} \boldsymbol{\rho}_a + \mathbf{r}_b^{ab}. \quad (6.127)$$

Inserting (6.120), we have that

$$z_b \mathbf{K}_b^{-1} \mathbf{q}_b = z_a \mathbf{C}_{ba} \mathbf{K}_a^{-1} \mathbf{q}_a + \mathbf{r}_b^{ab}. \quad (6.128)$$

We can then isolate for \mathbf{q}_b in terms of \mathbf{q}_a :

$$\mathbf{q}_b = \frac{z_a}{z_b} \mathbf{K}_b \mathbf{C}_{ba} \mathbf{K}_a^{-1} \mathbf{q}_a + \frac{1}{z_b} \mathbf{K}_b \mathbf{r}_b^{ab}. \quad (6.129)$$

Then, substituting z_b from (6.124), we have

$$\mathbf{q}_b = \frac{z_a}{z_b} \mathbf{K}_b \mathbf{C}_{ba} \left(\mathbf{1} + \frac{1}{d_a} \mathbf{r}_a^{ba} \mathbf{n}_a^T \right) \mathbf{K}_a^{-1} \mathbf{q}_a, \quad (6.130)$$

where we used that $\mathbf{r}_b^{ab} = -\mathbf{C}_{ba} \mathbf{r}_a^{ba}$. Finally, we can write

$$\mathbf{q}_b = \mathbf{K}_b \mathbf{H}_{ba} \mathbf{K}_a^{-1} \mathbf{q}_a, \quad (6.131)$$

where

$$\mathbf{H}_{ba} = \frac{z_a}{z_b} \mathbf{C}_{ba} \left(\mathbf{1} + \frac{1}{d_a} \mathbf{r}_a^{ba} \mathbf{n}_a^T \right) \quad (6.132)$$

is called the *homography matrix*. Since the factor z_a/z_b just scales \mathbf{q}_b , it can be dropped in practice owing to the fact that \mathbf{q}_b are homogeneous coordinates and the true pixel coordinates can always be recovered by dividing the first two entries by the third; doing so means that \mathbf{H}_{ba} is only a function of the pose change and the plane parameters.

It is worth noting that in the case of a pure rotation, $\mathbf{r}_a^{ba} = \mathbf{0}$, so that the homography matrix simplifies to

$$\mathbf{H}_{ba} = \mathbf{C}_{ba} \quad (6.133)$$

when the z_a/z_b factor is dropped.

The homography matrix is invertible and its inverse is given by

$$\mathbf{H}_{ba}^{-1} = \mathbf{H}_{ab} = \frac{z_b}{z_a} \mathbf{C}_{ab} \left(\mathbf{1} + \frac{1}{d_b} \mathbf{r}_b^{ab} \mathbf{n}_b^T \right). \quad (6.134)$$

This allows us to transform observations in the other direction.

6.4.2 Stereo Camera

Another common three-dimensional sensor is a stereo camera, which consists of two perspective cameras rigidly connected to one another with a known transformation between them. Figure 6.11 depicts one of the most common stereo configurations where the two cameras are separated along the x -axis by a *stereo baseline* of b . Unlike a single camera, it is possible to determine depth to a point from a stereo observation.

Midpoint Model

If we express the coordinates of the point, P , in \mathcal{F}_s as

$$\boldsymbol{\rho} = \mathbf{r}_s^{ps} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \quad (6.135)$$

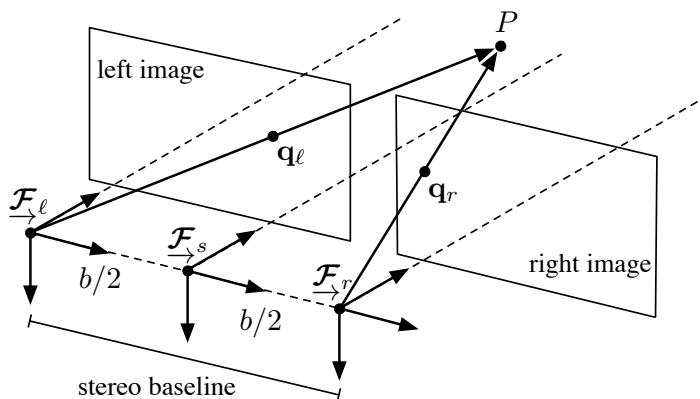
then the model for the left camera is

$$\begin{bmatrix} u_\ell \\ v_\ell \end{bmatrix} = \mathbf{P} \mathbf{K} \frac{1}{z} \begin{bmatrix} x + \frac{b}{2} \\ y \\ z \end{bmatrix}, \quad (6.136)$$

and the model for the right camera is

$$\begin{bmatrix} u_r \\ v_r \end{bmatrix} = \mathbf{P} \mathbf{K} \frac{1}{z} \begin{bmatrix} x - \frac{b}{2} \\ y \\ z \end{bmatrix}, \quad (6.137)$$

Figure 6.11
Stereo camera rig.
Two cameras are mounted pointing in the same direction but with a known separation of b along the x -axis. We choose the sensor frame to be located at the midpoint between the two cameras.



where we assume the two cameras have the same intrinsic parameter matrix. Stacking the two observations together, we can write the stereo camera model as

$$\begin{bmatrix} u_\ell \\ v_\ell \\ u_r \\ v_r \end{bmatrix} = \mathbf{s}(\boldsymbol{\rho}) = \underbrace{\begin{bmatrix} f_u & 0 & c_u & f_u \frac{b}{2} \\ 0 & f_v & c_v & 0 \\ f_u & 0 & c_u & -f_u \frac{b}{2} \\ 0 & f_v & c_v & 0 \end{bmatrix}}_{\mathbf{M}} \frac{1}{z} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \quad (6.138)$$

where \mathbf{M} is now a combined parameter matrix for the stereo rig. It is worth noting that \mathbf{M} is not invertible since two of its rows are the same. In fact, because of the stereo setup, the vertical coordinates of the two observations will always be the same; this corresponds to the fact that epipolar lines in this configuration are horizontal such that if a point is observed in one image, the observation in the other image can be found by searching along the line with the same vertical pixel coordinate. We can see this using the fundamental matrix constraint; for this stereo setup, we have $\mathbf{C}_{r\ell} = \mathbf{1}$ and $\mathbf{r}_r^{\ell r} = [-b \ 0 \ 0]$ so that the constraint is

$$[u_\ell \ v_\ell \ 1] \underbrace{\begin{bmatrix} f_u & 0 & 0 \\ 0 & f_v & 0 \\ c_u & c_v & 1 \end{bmatrix}}_{\mathbf{K}_\ell^T} \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & b \\ 0 & -b & 0 \end{bmatrix}}_{\mathbf{E}_{\ell r}} \underbrace{\begin{bmatrix} f_u & 0 & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{K}_r} \begin{bmatrix} u_r \\ v_r \\ 1 \end{bmatrix} = 0. \quad (6.139)$$

Multiplying this out, we see that it simplifies to $v_r = v_\ell$.

Left Model

We could also choose to locate the sensor frame at the left camera rather than the midpoint between the two cameras. In this case, the

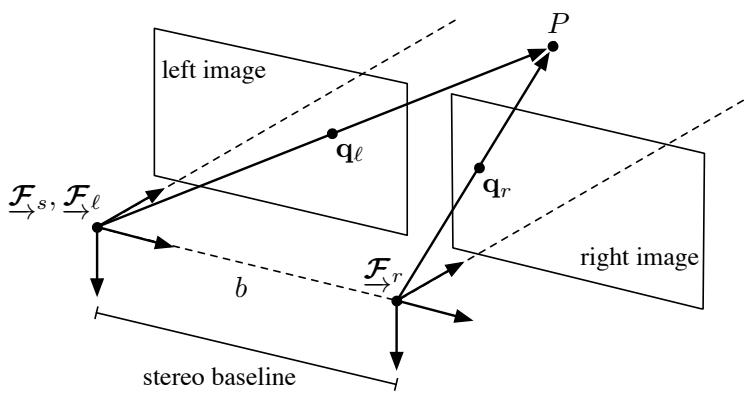


Figure 6.12
Alternate stereo model with the sensor frame located at the left camera.

camera model becomes

$$\begin{bmatrix} u_\ell \\ v_\ell \\ u_r \\ v_r \end{bmatrix} = \begin{bmatrix} f_u & 0 & c_u & 0 \\ 0 & f_v & c_v & 0 \\ f_u & 0 & c_u & -f_u b \\ 0 & f_v & c_v & 0 \end{bmatrix} \frac{1}{z} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}. \quad (6.140)$$

Typically, in this form, the v_r equation is dropped and the u_r equation is replaced with one for the *disparity*⁸, d , given by

$$d = u_\ell - u_r = \frac{1}{z} f_u b, \quad (6.141)$$

so that we can write

$$\begin{bmatrix} u_\ell \\ v_\ell \\ d \end{bmatrix} = \mathbf{s}(\boldsymbol{\rho}) = \begin{bmatrix} f_u & 0 & c_u & 0 \\ 0 & f_v & c_v & 0 \\ 0 & 0 & 0 & f_u b \end{bmatrix} \frac{1}{z} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \quad (6.142)$$

for the stereo model. This form has the appealing property that we are going from three point parameters, (x, y, z) , to three observations, (u_ℓ, v_ℓ, d) . A similar model can be developed for the right camera.

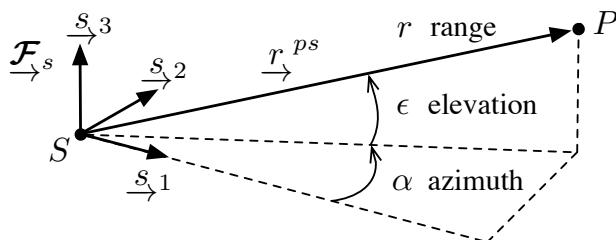
6.4.3 Range-Azimuth-Elevation

Some sensors, such as lidar (light detection and ranging), can be modelled as a *range-azimuth-elevation (RAE)*, which essentially observes a point, P , in spherical coordinates. For lidar, which can measure distance by reflecting laser pulses off a scene, the azimuth and elevation are the angles of the mirrors that are used to steer the laser beam and the range is the reported distance determined by time of flight. The geometry of this sensor type is depicted in Figure 6.13.

The coordinates of point P in the sensor frame, $\underline{\mathcal{F}}_s$, are

$$\boldsymbol{\rho} = \mathbf{r}_s^{ps} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}. \quad (6.143)$$

Figure 6.13 A range-azimuth-elevation sensor model observes a point P in spherical coordinates.



⁸ The disparity equation can be used as a one-dimensional stereo camera model, as we have already seen in the earlier chapter on nonlinear estimation.

These can also be written as

$$\boldsymbol{\rho} = \mathbf{C}_3^T(\alpha) \mathbf{C}_2^T(-\epsilon) \begin{bmatrix} r \\ 0 \\ 0 \end{bmatrix}, \quad (6.144)$$

where α is the azimuth, ϵ is the elevation, r is the range, and \mathbf{C}_i is the principal rotation about axis i . The elevation rotation indicated in Figure 6.13 is negative according to the right-hand rule. Inserting the principal-axis rotation formulas and multiplying out, we find that

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} r \cos \alpha \cos \epsilon \\ r \sin \alpha \cos \epsilon \\ r \sin \epsilon \end{bmatrix}, \quad (6.145)$$

which are the common spherical-coordinate expressions. Unfortunately, this is the inverse of the sensor model we desire. We can invert this expression to show that the RAE sensor model is

$$\begin{bmatrix} r \\ \alpha \\ \epsilon \end{bmatrix} = \mathbf{s}(\boldsymbol{\rho}) = \begin{bmatrix} \sqrt{x^2 + y^2 + z^2} \\ \tan^{-1}(y/x) \\ \sin^{-1}(z/\sqrt{x^2 + y^2 + z^2}) \end{bmatrix}. \quad (6.146)$$

In the case that the point P lies in the xy -plane, we have $z = 0$ and hence $\epsilon = 0$, so that the RAE model simplifies to the range-bearing model:

$$\begin{bmatrix} r \\ \alpha \end{bmatrix} = \mathbf{s}(\boldsymbol{\rho}) = \begin{bmatrix} \sqrt{x^2 + y^2} \\ \tan^{-1}(y/x) \end{bmatrix}, \quad (6.147)$$

which is commonly used in mobile robotics.

6.4.4 Inertial Measurement Unit

Another common sensor that functions in three-dimensional space is the *inertial measurement unit (IMU)*. An ideal IMU comprises three orthogonal linear accelerometers and three orthogonal rate gyros⁹. All quantities are measured in a sensor frame, $\underline{\mathcal{F}}_s$, which is typically not located at the vehicle frame, $\underline{\mathcal{F}}_v$, as shown in Figure 6.14.

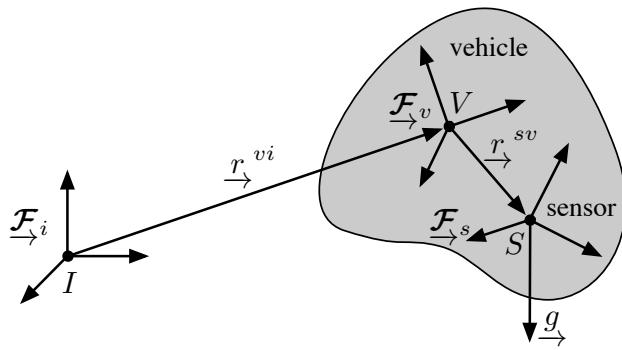
To model an IMU, we assume that the state of the vehicle can be captured by the quantities

$$\underbrace{\mathbf{r}_i^{vi}}_{\text{pose}}, \quad \underbrace{\mathbf{C}_{vi}}_{\text{pose}}, \quad \underbrace{\boldsymbol{\omega}_v^{vi}}_{\text{angular velocity}}, \quad \underbrace{\dot{\boldsymbol{\omega}}_v^{vi}}_{\text{angular acceleration}}, \quad (6.148)$$

and that we know the fixed pose change between the vehicle and sensor frames given by \mathbf{r}_v^{sv} and \mathbf{C}_{sv} , which is typically determined by calibration.

⁹ Typically, calibration is required, as the axes are never perfectly orthogonal due to manufacturing tolerances.

Figure 6.14 An inertial measurement unit has three linear accelerometers and three rate gyros that measure quantities in the sensor frame, which is typically not coincident with the vehicle frame.



The gyro sensor model is simpler than the accelerometers, so we will discuss this first. Essentially, the measured angular rates, ω , are the body rates of the vehicle, expressed in the sensor frame:

$$\omega = \mathbf{C}_{sv}\omega_v^{vi}. \quad (6.149)$$

This exploits the fact that the sensor frame is fixed with respect to the vehicle frame so that $\dot{\mathbf{C}}_{sv} = \mathbf{0}$.

Because accelerometers typically use test masses as part of the measurement principle, the resulting observations, \mathbf{a} , can be written as

$$\mathbf{a} = \mathbf{C}_{si}(\ddot{\mathbf{r}}_i^{si} - \mathbf{g}_i), \quad (6.150)$$

where $\ddot{\mathbf{r}}_i^{si}$ is the inertial acceleration of the sensor point, S , and \mathbf{g}_i is gravity. Notably, in freefall, the accelerometers will measure $\mathbf{a} = \mathbf{0}$, whereas at rest, they will measure only gravity (in the sensor frame). Unfortunately, this accelerometer model is not in terms of the vehicle state quantities that we identified above, and must be modified to account for the offset between the sensor and vehicle frames. To do this, we note that

$$\mathbf{r}_i^{si} = \mathbf{r}_i^{vi} + \mathbf{C}_{vi}^T \mathbf{r}_v^{sv}. \quad (6.151)$$

Differentiating twice (and using Poisson's equation from (6.45) and that $\dot{\mathbf{r}}_v^{sv} = \mathbf{0}$) provides

$$\ddot{\mathbf{r}}_i^{si} = \ddot{\mathbf{r}}_i^{vi} + \mathbf{C}_{vi}^T \dot{\boldsymbol{\omega}}_v^{vi\wedge} \mathbf{r}_v^{sv} + \mathbf{C}_{vi}^T \boldsymbol{\omega}_v^{vi\wedge} \boldsymbol{\omega}_v^{vi\wedge} \mathbf{r}_v^{sv}, \quad (6.152)$$

where the right-hand side is now in terms of our state quantities¹⁰ and known calibration parameters. Inserting this into (6.150) gives our final model for the accelerometers:

$$\mathbf{a} = \mathbf{C}_{sv} \left(\mathbf{C}_{vi} (\ddot{\mathbf{r}}_i^{vi} - \mathbf{g}_i) + \dot{\boldsymbol{\omega}}_v^{vi\wedge} \mathbf{r}_v^{sv} + \boldsymbol{\omega}_v^{vi\wedge} \boldsymbol{\omega}_v^{vi\wedge} \mathbf{r}_v^{sv} \right). \quad (6.153)$$

Naturally, if the offset between the sensor and vehicle frames, \mathbf{r}_v^{sv} , is sufficiently small, we may choose to neglect the last two terms.

¹⁰ If the angular acceleration quantity, $\dot{\boldsymbol{\omega}}_v^{vi}$, is not actually part of the state, it could be estimated from two or more measurements from the gyro at previous times.

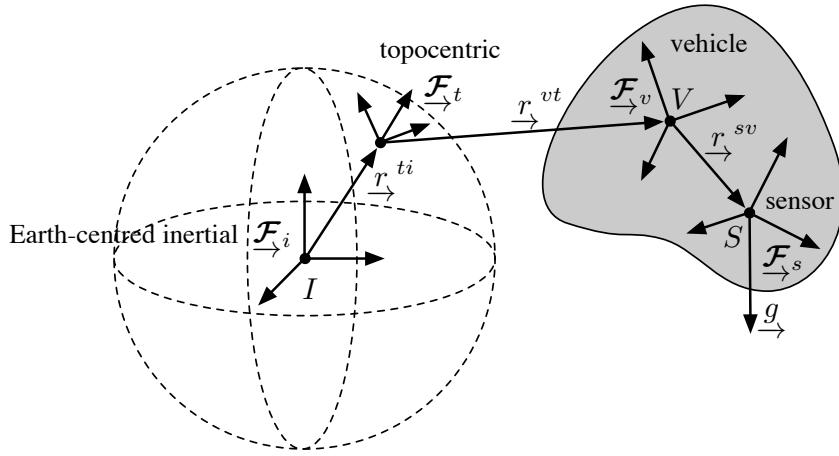


Figure 6.15 For high-performance inertial-measurement-unit applications, it is necessary to account for the rotation of the Earth, in which case a topocentric reference frame is located on the Earth's surface and an inertial frame is located at the Earth's center.

To summarize, we can stack the accelerometer and gyro models into the following combined IMU sensor model:

$$\begin{bmatrix} \mathbf{a} \\ \boldsymbol{\omega} \end{bmatrix} = \mathbf{s}(\mathbf{r}_i^{vi}, \mathbf{C}_{vi}, \boldsymbol{\omega}_v^{vi}, \dot{\boldsymbol{\omega}}_v^{vi}) \\ = \begin{bmatrix} \mathbf{C}_{sv} (\ddot{\mathbf{r}}_i^{vi} - \mathbf{g}_i) + \dot{\boldsymbol{\omega}}_v^{vi} \wedge \mathbf{r}_v^{sv} + \boldsymbol{\omega}_v^{vi} \wedge \dot{\boldsymbol{\omega}}_v^{vi} \wedge \mathbf{r}_v^{sv} \\ \mathbf{C}_{sv} \boldsymbol{\omega}_v^{vi} \end{bmatrix}, \quad (6.154)$$

where \mathbf{C}_{sv} and \mathbf{r}_v^{sv} are the (known) pose change between the vehicle and sensor frames and \mathbf{g}_i is gravity in the inertial frame.

For some high-performance inertial-measurement-unit applications, the above model is insufficient since it assumes an inertial reference frame can be located conveniently on the Earth's surface, for example. High-end IMU units, however, are sensitive enough to detect the rotation of the Earth, and thus a more elaborate model of the sensor is required. The typical setup is depicted in Figure 6.15, where the inertial frame is located at the Earth's center of mass (but not rotating) and then a convenient (non-inertial) reference frame (used to track the vehicle's motion) is located on the Earth's surface. This requires generalizing the sensor model to account for this more sophisticated setup (not shown).

6.5 Summary

The main take-away points from this chapter are as follows:

1. Objects that are able to rotate in three dimensions pose a problem for our state estimation techniques in the first part of the book. This is because we cannot, in general, use a vectorspace to describe the three-dimensional orientation of an object.

2. There are several ways to parameterize rotations (e.g., rotation matrix, Euler angles, unit-length quaternions). They all have advantages and disadvantages; some have singularities while the others have constraints. Our choice in this book is to favour the use of the rotation matrix since this is the quantity that is most commonly used to rotate vectors from one reference frame to another.
3. There are many different notational conventions in use in different fields (i.e., robotics, computer vision, aerospace). Coupled with the variety of rotational parameterizations, this can often lead to a source of miscommunication in practice. Our goal in this book is only to attempt to explain three-dimensional state estimation consistently and clearly in just one of the notational possibilities.

The next chapter will explore more deeply the mathematics of rotations and poses by introducing matrix Lie groups.

6.6 Exercises

- 6.6.1 Show that $\mathbf{u}^\wedge \mathbf{v} \equiv -\mathbf{v}^\wedge \mathbf{u}$ for any two 3×1 columns \mathbf{u} and \mathbf{v} .
- 6.6.2 Show that $\mathbf{C}^{-1} = \mathbf{C}^T$ starting from

$$\mathbf{C} = \cos \theta \mathbf{1} + (1 - \cos \theta) \mathbf{a} \mathbf{a}^T + \sin \theta \mathbf{a}^\wedge.$$

- 6.6.3 Show that $(\mathbf{C}\mathbf{v})^\wedge \equiv \mathbf{C}\mathbf{v}^\wedge \mathbf{C}^T$ for any 3×1 column \mathbf{v} and rotation matrix, \mathbf{C} .
- 6.6.4 Show that

$$\dot{\mathbf{T}}_{iv} = \mathbf{T}_{iv} \begin{bmatrix} 0 & -v\kappa & 0 & v \\ v\kappa & 0 & -v\tau & 0 \\ 0 & v\tau & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

- 6.6.5 Show that if we constrain the motion to the xy -plane, the Frenet-Serret equations simplify to

$$\begin{aligned} \dot{x} &= v \cos \theta, \\ \dot{y} &= v \sin \theta, \\ \dot{\theta} &= \omega, \end{aligned}$$

where $\omega = v\kappa$.

- 6.6.6 Show that for the single-camera model, straight lines in Euclidean space project to straight lines in image space.
- 6.6.7 Show directly that the inverse of the homography matrix

$$\mathbf{H}_{ba} = \frac{z_a}{z_b} \mathbf{C}_{ba} \left(\mathbf{1} + \frac{1}{d_a} \mathbf{r}_a^{ba} \mathbf{n}_a^T \right),$$

is

$$\mathbf{H}_{ba}^{-1} = \frac{z_b}{z_a} \mathbf{C}_{ab} \left(\mathbf{1} + \frac{1}{d_b} \mathbf{r}_b^{ab} \mathbf{n}_b^T \right).$$

- 6.6.8 Work out the stereo camera model for the case when the sensor frame is located at the right camera instead of the left or the midpoint.
- 6.6.9 Work out the inverse of the left stereo camera model. In other words, how can we go from (u_ℓ, v_ℓ, d) back to the point coordinates, (x, y, z) ?
- 6.6.10 Work out an IMU model for the situation depicted in Figure 6.15, where it is necessary to account for the rotation of the Earth about its axis.

7

Matrix Lie Groups

We have already introduced *rotations* and *poses* in the previous chapter on three-dimensional geometry. In this chapter, we look more deeply into the nature of these quantities. It turns out that rotations are quite different from the usual vector quantities with which we are familiar. The set of rotations is not a *vectorspace* in the sense of linear algebra. However, rotations do form another mathematical object called a *non-commutative group*, which possesses some, but not all, of the usual vectorspace properties.

We will focus our efforts in this chapter on two sets known as *matrix Lie groups*. Stillwell (2008) provides an accessible account of Lie theory, and Chirikjian (2009) is an excellent reference from the robotics perspective.

Marius Sophus Lie (1842-1899) was a Norwegian mathematician. He largely created the theory of continuous symmetry and applied it to the study of geometry and differential equations.

7.1 Geometry

We will work with two specific matrix Lie groups in this chapter: the *special orthogonal group*, denoted $SO(3)$, which can represent rotations, and the *special Euclidean group*, $SE(3)$, which can represent poses.

7.1.1 Special Orthogonal and Special Euclidean Groups

The *special orthogonal group*, representing rotations, is simply the set of valid rotation matrices:

$$SO(3) = \{\mathbf{C} \in \mathbb{R}^{3 \times 3} \mid \mathbf{C}\mathbf{C}^T = \mathbf{1}, \det \mathbf{C} = 1\}. \quad (7.1)$$

The $\mathbf{C}\mathbf{C}^T = \mathbf{1}$ orthogonality condition is needed to impose six constraints on the nine-parameter rotation matrix, thereby reducing the number of degrees of freedom to three. Noticing that

$$(\det \mathbf{C})^2 = \det (\mathbf{C}\mathbf{C}^T) = \det \mathbf{1} = 1, \quad (7.2)$$

we have that

$$\det \mathbf{C} = \pm 1, \quad (7.3)$$

allowing for two possibilities. Choosing $\det \mathbf{C} = 1$ ensures that we have a *proper rotation*¹.

Although the set of all matrices can be shown to be a vectorspace, $SO(3)$ is not a valid subspace². For example, $SO(3)$ is not closed under addition, so adding two rotation matrices does not result in a valid rotation matrix:

$$\mathbf{C}_1, \mathbf{C}_2 \in SO(3) \nRightarrow \mathbf{C}_1 + \mathbf{C}_2 \in SO(3). \quad (7.4)$$

Also, the zero matrix is not a valid rotation matrix: $\mathbf{0} \notin SO(3)$. Without these properties (and some others), $SO(3)$ cannot be a vectorspace (at least not a subspace of $\mathbb{R}^{3 \times 3}$).

The *special Euclidean group*, representing poses (i.e., translation and rotation), is simply the set of valid transformation matrices:

$$SE(3) = \left\{ \mathbf{T} = \begin{bmatrix} \mathbf{C} & \mathbf{r} \\ \mathbf{0}^T & 1 \end{bmatrix} \in \mathbb{R}^{4 \times 4} \mid \mathbf{C} \in SO(3), \mathbf{r} \in \mathbb{R}^3 \right\}. \quad (7.5)$$

By similar arguments to $SO(3)$, we can show that $SE(3)$ is not a vectorspace (at least not a subspace of $\mathbb{R}^{4 \times 4}$).

While $SO(3)$ and $SE(3)$ are not vectorspaces, they can be shown to be *matrix Lie groups*³. We next show what this means. In mathematics, a *group* is a set of elements together with an operation that combines any two of its elements to form a third element also in the set, while satisfying four conditions called the group axioms, namely closure, associativity, identity, and invertibility. A *Lie group* is a group that is also a *differential manifold*, with the property that the group operations are smooth⁴. A *matrix Lie group* further specifies that the elements of the group are matrices, the combination operation is matrix multiplication, and the inversion operation is matrix inversion.

The four group properties are then as shown in Table 7.1.1 for our two candidate matrix Lie groups. Closure for $SO(3)$ actually follows directly from Euler's rotation theorem, which says a compounding of rotations can be replaced by a single rotation. Or, we can note that

$$\begin{aligned} (\mathbf{C}_1 \mathbf{C}_2) (\mathbf{C}_1 \mathbf{C}_2)^T &= \mathbf{C}_1 \underbrace{\mathbf{C}_2 \mathbf{C}_2^T}_{1} \mathbf{C}_1^T = \underbrace{\mathbf{C}_1 \mathbf{C}_1^T}_{1} = \mathbf{1}, \\ \det(\mathbf{C}_1 \mathbf{C}_2) &= \underbrace{\det(\mathbf{C}_1)}_{1} \underbrace{\det(\mathbf{C}_2)}_{1} = 1, \end{aligned} \quad (7.6)$$

¹ There is another case in which $\det \mathbf{C} = -1$, sometimes called an *improper rotation* or *rotary reflection*, but we shall not be concerned with it here.

² A subspace of a vectorspace is also a vectorspace.

³ They are actually *non-Abelian* (or non-commutative) groups since the order in which we compound elements matters.

⁴ Smoothness implies that we can use differential calculus on the manifold; or, roughly, if we change the input to any group operation by a little bit, the output will only change by a little bit.

property	$SO(3)$	$SE(3)$
closure	$\mathbf{C}_1, \mathbf{C}_2 \in SO(3)$ $\Rightarrow \mathbf{C}_1 \mathbf{C}_2 \in SO(3)$	$\mathbf{T}_1, \mathbf{T}_2 \in SE(3)$ $\Rightarrow \mathbf{T}_1 \mathbf{T}_2 \in SE(3)$
associativity	$\mathbf{C}_1 (\mathbf{C}_2 \mathbf{C}_3) = (\mathbf{C}_1 \mathbf{C}_2) \mathbf{C}_3$ $= \mathbf{C}_1 \mathbf{C}_2 \mathbf{C}_3$	$\mathbf{T}_1 (\mathbf{T}_2 \mathbf{T}_3) = (\mathbf{T}_1 \mathbf{T}_2) \mathbf{T}_3$ $= \mathbf{T}_1 \mathbf{T}_2 \mathbf{T}_3$
identity	$\mathbf{C}, \mathbf{1} \in SO(3)$ $\Rightarrow \mathbf{C}\mathbf{1} = \mathbf{1}\mathbf{C} = \mathbf{C}$	$\mathbf{T}, \mathbf{1} \in SE(3)$ $\Rightarrow \mathbf{T}\mathbf{1} = \mathbf{1}\mathbf{T} = \mathbf{T}$
invertibility	$\mathbf{C} \in SO(3)$ $\Rightarrow \mathbf{C}^{-1} \in SO(3)$	$\mathbf{T} \in SE(3)$ $\Rightarrow \mathbf{T}^{-1} \in SE(3)$

Table 7.1 Matrix Lie group properties for $SO(3)$ (rotations) and $SE(3)$ (poses).

such that $\mathbf{C}_1 \mathbf{C}_2 \in SO(3)$ if $\mathbf{C}_1, \mathbf{C}_2 \in SO(3)$. Closure for $SE(3)$ can be seen simply by multiplying,

$$\mathbf{T}_1 \mathbf{T}_2 = \begin{bmatrix} \mathbf{C}_1 & \mathbf{r}_1 \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{C}_2 & \mathbf{r}_2 \\ \mathbf{0}^T & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{C}_1 \mathbf{C}_2 & \mathbf{C}_1 \mathbf{r}_2 + \mathbf{r}_1 \\ \mathbf{0}^T & 1 \end{bmatrix} \in SE(3), \quad (7.7)$$

since $\mathbf{C}_1 \mathbf{C}_2 \in SO(3)$ and $\mathbf{C}_1 \mathbf{r}_2 + \mathbf{r}_1 \in \mathbb{R}^3$. Associativity follows for both groups from the properties of matrix multiplication⁵. The identity matrix is the identity element of both groups, which again follows from the properties of matrix multiplication. Finally, since $\mathbf{C}^{-1} = \mathbf{C}^T$, which follows from $\mathbf{C}\mathbf{C}^T = \mathbf{1}$, we know that the inverse of an element of $SO(3)$ is still in $SO(3)$. This can be seen through

$$\begin{aligned} (\mathbf{C}^{-1}) (\mathbf{C}^{-1})^T &= (\mathbf{C}^T) (\mathbf{C}^T)^T = \underbrace{\mathbf{C}^T \mathbf{C}}_1 = \mathbf{1}, \\ \det(\mathbf{C}^{-1}) &= \det(\mathbf{C}^T) = \underbrace{\det \mathbf{C}}_1 = 1. \end{aligned} \quad (7.8)$$

The inverse of an element of $SE(3)$ is

$$\mathbf{T}^{-1} = \begin{bmatrix} \mathbf{C} & \mathbf{r} \\ \mathbf{0}^T & 1 \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{C}^T & -\mathbf{C}^T \mathbf{r} \\ \mathbf{0}^T & 1 \end{bmatrix} \in SE(3); \quad (7.9)$$

since $\mathbf{C}^T \in SO(3)$, $-\mathbf{C}^T \mathbf{r} \in \mathbb{R}^3$, so this also holds. Other than the smoothness criterion, this establishes $SO(3)$ and $SE(3)$ as matrix Lie groups.

7.1.2 Lie Algebras

With every matrix Lie group is associated a *Lie algebra*, which consists of a vectorspace⁶, \mathbb{V} , over some field⁷, \mathbb{F} , together with a binary operation, $[\cdot, \cdot]$, called the *Lie bracket* (of the algebra) and that satisfies four properties:

⁵ The set of all real matrices can be shown to be an *algebra* and associativity of matrix multiplication is a required property.

⁶ We can take this to be a subspace of the square real matrices, which is a vectorspace.

⁷ We can take this to be the field of real numbers, \mathbb{R} .

$$\begin{aligned}
\text{closure: } & [\mathbf{X}, \mathbf{Y}] \in \mathbb{V}, \\
\text{bilinearity: } & [a\mathbf{X} + b\mathbf{Y}, \mathbf{Z}] = a[\mathbf{X}, \mathbf{Z}] + b[\mathbf{Y}, \mathbf{Z}], \\
& [\mathbf{Z}, a\mathbf{X} + b\mathbf{Y}] = a[\mathbf{Z}, \mathbf{X}] + b[\mathbf{Z}, \mathbf{Y}], \\
\text{alternating: } & [\mathbf{X}, \mathbf{X}] = \mathbf{0}, \\
\text{Jacobi identity: } & [\mathbf{X}, [\mathbf{Y}, \mathbf{Z}]] + [\mathbf{Y}, [\mathbf{Z}, \mathbf{X}]] + [\mathbf{Z}, [\mathbf{X}, \mathbf{Y}]] = \mathbf{0},
\end{aligned}$$

for all $\mathbf{X}, \mathbf{Y}, \mathbf{Z} \in \mathbb{V}$ and $a, b \in \mathbb{F}$. The vectorspace of a Lie algebra is the *tangent space* of the associated Lie group at the identity element of the group, and it completely captures the local structure of the group.

Rotations

The Lie algebra associated with $SO(3)$ is given by

$$\begin{aligned}
\text{vectorspace: } & \mathfrak{so}(3) = \{\Phi = \phi^\wedge \in \mathbb{R}^{3 \times 3} \mid \phi \in \mathbb{R}^3\}, \\
\text{field: } & \mathbb{R}, \\
\text{Lie bracket: } & [\Phi_1, \Phi_2] = \Phi_1 \Phi_2 - \Phi_2 \Phi_1,
\end{aligned}$$

where

$$\phi^\wedge = \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \end{bmatrix}^\wedge = \begin{bmatrix} 0 & -\phi_3 & \phi_2 \\ \phi_3 & 0 & -\phi_1 \\ -\phi_2 & \phi_1 & 0 \end{bmatrix} \in \mathbb{R}^{3 \times 3}, \quad \phi \in \mathbb{R}^3. \quad (7.10)$$

We already saw this linear, skew-symmetric operator in the previous chapter during our discussion of cross products and rotations. Later, we will also make use of the inverse of this operator, denoted $(\cdot)^\vee$, so that

$$\Phi = \phi^\wedge \Rightarrow \phi = \Phi^\vee. \quad (7.11)$$

We will omit proving that $\mathfrak{so}(3)$ is a vectorspace, but will briefly show that the four Lie bracket properties hold. Let $\Phi, \Phi_1 = \phi_1^\wedge, \Phi_2 = \phi_2^\wedge \in \mathfrak{so}(3)$. Then, for the closure property, we have

$$[\Phi_1, \Phi_2] = \Phi_1 \Phi_2 - \Phi_2 \Phi_1 = \underbrace{\phi_1^\wedge \phi_2^\wedge}_{\in \mathbb{R}^3} - \underbrace{\phi_2^\wedge \phi_1^\wedge}_{\in \mathbb{R}^3} = (\underbrace{\phi_1^\wedge \phi_2^\wedge}_{\in \mathbb{R}^3})^\wedge \in \mathfrak{so}(3). \quad (7.12)$$

Bilinearity follows directly from the fact that $(\cdot)^\wedge$ is a linear operator. The alternating property can be seen easily through

$$[\Phi, \Phi] = \Phi \Phi - \Phi \Phi = \mathbf{0} \in \mathfrak{so}(3). \quad (7.13)$$

Finally, the Jacobi identity can be verified by substituting and applying the definition of the Lie bracket. Informally, we will refer to $\mathfrak{so}(3)$ as the Lie algebra, although technically this is only the associated vectorspace.

Poses

The Lie algebra associated with $SE(3)$ is given by

Carl Gustav Jacob Jacobi (1804-1851) was a German mathematician who made fundamental contributions to elliptic functions, dynamics, differential equations, and number theory.

vectorspace: $\mathfrak{se}(3) = \{\boldsymbol{\Xi} = \boldsymbol{\xi}^\wedge \in \mathbb{R}^{4 \times 4} \mid \boldsymbol{\xi} \in \mathbb{R}^6\}$,
field: \mathbb{R} ,
Lie bracket: $[\boldsymbol{\Xi}_1, \boldsymbol{\Xi}_2] = \boldsymbol{\Xi}_1 \boldsymbol{\Xi}_2 - \boldsymbol{\Xi}_2 \boldsymbol{\Xi}_1$,

where

$$\boldsymbol{\xi}^\wedge = \begin{bmatrix} \boldsymbol{\rho} \\ \boldsymbol{\phi} \end{bmatrix}^\wedge = \begin{bmatrix} \boldsymbol{\phi}^\wedge & \boldsymbol{\rho} \\ \mathbf{0}^T & 0 \end{bmatrix} \in \mathbb{R}^{4 \times 4}, \quad \boldsymbol{\rho}, \boldsymbol{\phi} \in \mathbb{R}^3. \quad (7.14)$$

This is an overloading of the $(\cdot)^\wedge$ operator (Murray et al., 1994) from before to take elements of \mathbb{R}^6 and turn them into elements of $\mathbb{R}^{4 \times 4}$; it is still linear. We will also make use of the inverse of this operator, denoted $(\cdot)^\vee$, so that

$$\boldsymbol{\Xi} = \boldsymbol{\xi}^\wedge \Rightarrow \boldsymbol{\xi} = \boldsymbol{\Xi}^\vee. \quad (7.15)$$

Again, we will omit showing that $\mathfrak{se}(3)$ is a vectorspace, but will briefly show that the four Lie bracket properties hold. Let $\boldsymbol{\Xi}, \boldsymbol{\Xi}_1 = \boldsymbol{\xi}_1^\wedge, \boldsymbol{\Xi}_2 = \boldsymbol{\xi}_2^\wedge \in \mathfrak{se}(3)$. Then, for the closure property, we have

$$[\boldsymbol{\Xi}_1, \boldsymbol{\Xi}_2] = \boldsymbol{\Xi}_1 \boldsymbol{\Xi}_2 - \boldsymbol{\Xi}_2 \boldsymbol{\Xi}_1 = \boldsymbol{\xi}_1^\wedge \boldsymbol{\xi}_2^\wedge - \boldsymbol{\xi}_2^\wedge \boldsymbol{\xi}_1^\wedge = \underbrace{\left(\boldsymbol{\xi}_1^\wedge \boldsymbol{\xi}_2^\wedge \right)}_{\in \mathbb{R}^6}^\wedge \in \mathfrak{se}(3), \quad (7.16)$$

where

$$\boldsymbol{\xi}^\wedge = \begin{bmatrix} \boldsymbol{\rho} \\ \boldsymbol{\phi} \end{bmatrix}^\wedge = \begin{bmatrix} \boldsymbol{\phi}^\wedge & \boldsymbol{\rho}^\wedge \\ \mathbf{0} & \boldsymbol{\phi}^\wedge \end{bmatrix} \in \mathbb{R}^{6 \times 6}, \quad \boldsymbol{\rho}, \boldsymbol{\phi} \in \mathbb{R}^3. \quad (7.17)$$

Bilinearity follows directly from the fact that $(\cdot)^\wedge$ is a linear operator. The alternating property can be seen easily through

$$[\boldsymbol{\Xi}, \boldsymbol{\Xi}] = \boldsymbol{\Xi} \boldsymbol{\Xi} - \boldsymbol{\Xi} \boldsymbol{\Xi} = \mathbf{0} \in \mathfrak{se}(3). \quad (7.18)$$

Finally, the Jacobi identity can be verified by substituting and applying the definition of the Lie bracket. Again, we will refer to $\mathfrak{se}(3)$ as the Lie algebra, although technically, this is only the associated vectorspace.

In the next section, we will make clear the relationships between our matrix Lie groups and their associated Lie algebras:

$$\begin{aligned} SO(3) &\leftrightarrow \mathfrak{so}(3), \\ SE(3) &\leftrightarrow \mathfrak{se}(3). \end{aligned}$$

For this we require the *exponential map*.

7.1.3 Exponential Map

It turns out that the *exponential map* is the key to relating a matrix Lie group to its associated Lie algebra. The matrix exponential is given by

$$\exp(\mathbf{A}) = \mathbf{1} + \mathbf{A} + \frac{1}{2!} \mathbf{A}^2 + \frac{1}{3!} \mathbf{A}^3 + \cdots = \sum_{n=0}^{\infty} \frac{1}{n!} \mathbf{A}^n, \quad (7.19)$$

where $\mathbf{A} \in \mathbb{R}^{M \times M}$ is a square matrix. There is also a matrix logarithm:

$$\ln(\mathbf{A}) = \sum_{n=1}^{\infty} \frac{(-1)^{n-1}}{n} (\mathbf{A} - \mathbf{1})^n. \quad (7.20)$$

Rotations

For rotations, we can relate elements of $SO(3)$ to elements of $\mathfrak{so}(3)$ through the exponential map:

$$\mathbf{C} = \exp(\phi^\wedge) = \sum_{n=0}^{\infty} \frac{1}{n!} (\phi^\wedge)^n, \quad (7.21)$$

where $\mathbf{C} \in SO(3)$ and $\phi \in \mathbb{R}^3$ (and hence $\phi^\wedge \in \mathfrak{so}(3)$). We can also go in the other direction (but not uniquely) using

$$\phi = \ln(\mathbf{C})^\vee. \quad (7.22)$$

Mathematically, the exponential map from $\mathfrak{so}(3)$ to $SO(3)$ is *surjective-only* (or *surjective/onto* and *non-injective/many-to-one*). This means that we can generate every element of $SO(3)$ from multiple elements of $\mathfrak{so}(3)$ ⁸.

It is useful to examine the surjective-only property a little deeper. We begin by working out the forwards (exponential) mapping in closed form to go from a $\phi \in \mathbb{R}^3$ to a $\mathbf{C} \in SO(3)$. Let $\phi = \phi \mathbf{a}$, where $\phi = |\phi|$ is the angle of rotation and $\mathbf{a} = \phi/\phi$ is the unit-length axis of rotation. For the matrix exponential, we then have

$$\begin{aligned} \exp(\phi^\wedge) &= \exp(\phi \mathbf{a}^\wedge) \\ &= \underbrace{\mathbf{1}_{\mathbf{aa}^T - \mathbf{a}^\wedge \mathbf{a}^\wedge}}_{\mathbf{aa}^T - \mathbf{a}^\wedge \mathbf{a}^\wedge} + \phi \mathbf{a}^\wedge + \frac{1}{2!} \phi^2 \mathbf{a}^\wedge \mathbf{a}^\wedge + \frac{1}{3!} \phi^3 \underbrace{\mathbf{a}^\wedge \mathbf{a}^\wedge \mathbf{a}^\wedge}_{-\mathbf{a}^\wedge} \\ &\quad + \frac{1}{4!} \phi^4 \underbrace{\mathbf{a}^\wedge \mathbf{a}^\wedge \mathbf{a}^\wedge \mathbf{a}^\wedge}_{-\mathbf{a}^\wedge \mathbf{a}^\wedge} - \dots \\ &= \mathbf{aa}^T + \underbrace{\left(\phi - \frac{1}{3!} \phi^3 + \frac{1}{5!} \phi^5 - \dots \right)}_{\sin \phi} \mathbf{a}^\wedge \\ &\quad - \underbrace{\left(1 - \frac{1}{2!} \phi^2 + \frac{1}{4!} \phi^4 - \dots \right)}_{\cos \phi} \underbrace{\mathbf{a}^\wedge \mathbf{a}^\wedge}_{-\mathbf{1} + \mathbf{aa}^T} \\ &= \underbrace{\cos \phi \mathbf{1} + (1 - \cos \phi) \mathbf{aa}^T + \sin \phi \mathbf{a}^\wedge}_{\mathbf{C}}, \end{aligned} \quad (7.23)$$

⁸ The many-to-one mapping property is related to the concept of singularities (or nonuniqueness) in rotation parameterizations. We know that every three-parameter representation of a rotation cannot represent orientation both globally and uniquely. Nonuniqueness implies that given a \mathbf{C} , we cannot uniquely find a single $\phi \in \mathbb{R}^3$ that generated it; there is an infinite number of them.

which we see is the canonical axis-angle form of a rotation matrix presented earlier. We have used the useful identities (for unit-length \mathbf{a}),

$$\mathbf{a}^\wedge \mathbf{a}^\wedge \equiv -\mathbf{1} + \mathbf{a}\mathbf{a}^T, \quad (7.24a)$$

$$\mathbf{a}^\wedge \mathbf{a}^\wedge \mathbf{a}^\wedge \equiv -\mathbf{a}^\wedge, \quad (7.24b)$$

the proofs of which are left to the reader. This shows that every $\phi \in \mathbb{R}^3$ will generate a valid $\mathbf{C} \in SO(3)$. It also shows that if we add a multiple of 2π to the angle of rotation, we will generate the same \mathbf{C} . In detail, we have

$$\mathbf{C} = \exp((\phi + 2\pi m) \mathbf{a}^\wedge), \quad (7.25)$$

with m any positive integer, since $\cos(\phi + 2\pi m) = \cos \phi$ and $\sin(\phi + 2\pi m) = \sin \phi$. If we limit the angle of rotation, $|\phi| < \pi$, of the input, then each \mathbf{C} can only be generated by one ϕ .

Additionally, we would like to show that every $\mathbf{C} \in SO(3)$ can be generated by some $\phi \in \mathbb{R}^3$, and for that we need the inverse (logarithmic) mapping: $\phi = \ln(\mathbf{C})^\vee$. We can also work this out in closed form. Since a rotation matrix applied to its own axis does not alter the axis,

$$\mathbf{C}\mathbf{a} = \mathbf{a}, \quad (7.26)$$

this implies that \mathbf{a} is a (unit-length) eigenvector of \mathbf{C} corresponding to an eigenvalue of 1. Thus, by solving the eigenproblem associated with \mathbf{C} , we can find \mathbf{a}^9 . The angle can be found by exploiting the trace (sum of the diagonal elements) of a rotation matrix:

$$\begin{aligned} \text{tr}(\mathbf{C}) &= \text{tr}(\cos \phi \mathbf{1} + (1 - \cos \phi)\mathbf{a}\mathbf{a}^T + \sin \phi \mathbf{a}^\wedge) \\ &= \cos \phi \underbrace{\text{tr}(\mathbf{1})}_{3} + (1 - \cos \phi) \underbrace{\text{tr}(\mathbf{a}\mathbf{a}^T)}_{\mathbf{a}^T\mathbf{a}=1} + \sin \phi \underbrace{\text{tr}(\mathbf{a}^\wedge)}_0 = 2 \cos \phi + 1. \end{aligned} \quad (7.27)$$

Solving, we have

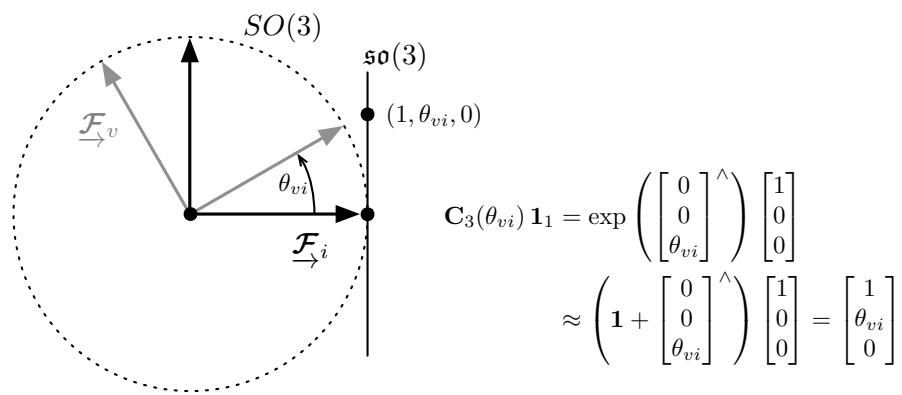
$$\phi = \cos^{-1} \left(\frac{\text{tr}(\mathbf{C}) - 1}{2} \right) + 2\pi m, \quad (7.28)$$

which indicates there are many solutions for ϕ . By convention, we will pick the one such that $|\phi| < \pi$. To complete the process, we combine \mathbf{a} and ϕ according to $\phi = \phi \mathbf{a}$. It is noteworthy that there is an ambiguity in the sign of ϕ since $\cos \phi$ is an even function; we can test for the correct one by going the other way to see that ϕ produces the correct \mathbf{C} and if not reversing the sign of ϕ . This shows that every $\mathbf{C} \in SO(3)$ can be built from at least one $\phi \in \mathbb{R}^3$.

Figure 7.1 provides a simple example of the relationship between the

⁹ There are some subtleties that occur when there is more than one eigenvalue equal to 1. E.g., $\mathbf{C} = \mathbf{1}$, whereupon \mathbf{a} is not unique and can be any unit vector.

Figure 7.1
 Example of the relationship between the Lie group and Lie algebra for the case of rotation constrained to the plane. In a small neighbourhood around the $\theta_{vi} = 0$ point, the vectorspace associated with the Lie algebra is a line tangent to the circle.



Lie group and Lie algebra for the case of rotation constrained to the plane. We see that in a neighbourhood near the zero-rotation point, $\theta_{vi} = 0$, the Lie algebra vectorspace is just a line that is tangent to the circle of rotation. We see that, indeed, near zero rotation, the Lie algebra captures the local structure of the Lie group. It should be pointed out that this example is constrained to the plane (i.e., a single rotational degree of freedom), but in general the dimension of the Lie algebra vectorspace is three. Put another way, the line in the figure is a one-dimensional subspace of the full three-dimensional Lie algebra vectorspace.

Connecting rotation matrices with the exponential map makes it easy to show that $\det(\mathbf{C}) = 1$ using *Jacobi's formula*, which, for a general square complex matrix, \mathbf{A} , says

$$\det(\exp(\mathbf{A})) = \exp(\text{tr}(\mathbf{A})). \quad (7.29)$$

In the case of rotations, we have

$$\det(\mathbf{C}) = \det(\exp(\phi^\wedge)) = \exp(\text{tr}(\phi^\wedge)) = \exp(0) = 1, \quad (7.30)$$

since ϕ^\wedge is skew-symmetric and therefore has zeros on its diagonal, making its trace zero.

Poses

For poses, we can relate elements of $SE(3)$ to elements of $\mathfrak{se}(3)$, again through the exponential map:

$$\mathbf{T} = \exp(\boldsymbol{\xi}^\wedge) = \sum_{n=0}^{\infty} \frac{1}{n!} (\boldsymbol{\xi}^\wedge)^n, \quad (7.31)$$

where $\mathbf{T} \in SE(3)$ and $\boldsymbol{\xi} \in \mathbb{R}^6$ (and hence $\boldsymbol{\xi}^\wedge \in \mathfrak{se}(3)$). We can also go in the other direction¹⁰ using

$$\boldsymbol{\xi} = \ln(\mathbf{T})^\vee. \quad (7.32)$$

¹⁰ Again, not uniquely.

The exponential map from $\mathfrak{se}(3)$ to $SE(3)$ is also surjective-only: every $\mathbf{T} \in SE(3)$ can be generated by many $\boldsymbol{\xi} \in \mathbb{R}^6$.

To show the surjective-only property of the exponential map, we first examine the forwards direction. Starting with $\boldsymbol{\xi} = \begin{bmatrix} \boldsymbol{\rho} \\ \phi \end{bmatrix} \in \mathbb{R}^6$, we have

$$\begin{aligned} \exp(\boldsymbol{\xi}^\wedge) &= \sum_{n=0}^{\infty} \frac{1}{n!} (\boldsymbol{\xi}^\wedge)^n \\ &= \sum_{n=0}^{\infty} \frac{1}{n!} \left(\begin{bmatrix} \boldsymbol{\rho} \\ \phi \end{bmatrix} \right)^\wedge \\ &= \sum_{n=0}^{\infty} \frac{1}{n!} \begin{bmatrix} \boldsymbol{\phi}^\wedge & \boldsymbol{\rho} \\ \mathbf{0}^T & 0 \end{bmatrix}^n \\ &= \begin{bmatrix} \sum_{n=0}^{\infty} \frac{1}{n!} (\boldsymbol{\phi}^\wedge)^n & \left(\sum_{n=0}^{\infty} \frac{1}{(n+1)!} (\boldsymbol{\phi}^\wedge)^n \right) \boldsymbol{\rho} \\ \mathbf{0}^T & 1 \end{bmatrix} \\ &= \underbrace{\begin{bmatrix} \mathbf{C} & \mathbf{r} \\ \mathbf{0}^T & 1 \end{bmatrix}}_{\mathbf{T}} \in SE(3), \end{aligned} \quad (7.33)$$

where

$$\mathbf{r} = \mathbf{J}\boldsymbol{\rho} \in \mathbb{R}^3, \quad \mathbf{J} = \sum_{n=0}^{\infty} \frac{1}{(n+1)!} (\boldsymbol{\phi}^\wedge)^n. \quad (7.34)$$

This shows that every $\boldsymbol{\xi} \in \mathbb{R}^6$ will generate a valid $\mathbf{T} \in SE(3)$. We will discuss the matrix, \mathbf{J} , in greater detail just below. Figure 7.2 provides

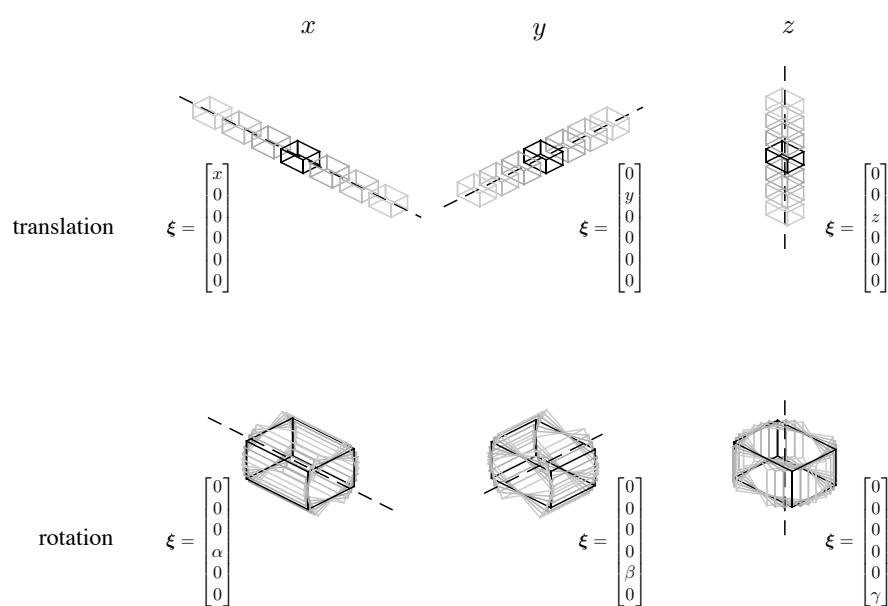


Figure 7.2 By varying each of the components of $\boldsymbol{\xi}$, constructing $\mathbf{T} = \exp(\boldsymbol{\xi}^\wedge)$, and then using this to transform the points composing the corners of a rectangular prism, we see that the prism's pose can be translated and rotated. Combining these basic movements can result in any arbitrary pose change of the prism.

a visualization of how each of the six components of ξ can be varied to alter the pose of a rectangular prism. By combining these basic translations and rotations, an arbitrary pose change can be achieved.

Next we would like to go in the inverse direction. Starting with $\mathbf{T} = \begin{bmatrix} \mathbf{C} & \mathbf{r} \\ \mathbf{0}^T & 1 \end{bmatrix}$, we want to show this can be generated by some $\xi = \begin{bmatrix} \rho \\ \phi \end{bmatrix} \in \mathbb{R}^6$; we need the inverse mapping, $\xi = \ln(\mathbf{T})^\vee$. We have already seen how to go from $\mathbf{C} \in SO(3)$ to $\phi \in \mathbb{R}^3$ in the last section. Next, we can compute

$$\rho = \mathbf{J}^{-1}\mathbf{r}, \quad (7.35)$$

where \mathbf{J} is built from ϕ (already computed). Finally, we assemble $\xi \in \mathbb{R}^6$ from $\rho, \phi \in \mathbb{R}^3$. This shows that every $\mathbf{T} \in SE(3)$ can be generated by at least one $\xi \in \mathbb{R}^6$.

Jacobian

The matrix, \mathbf{J} , described just above, plays an important role in allowing us to convert the translation component of pose in $\mathfrak{se}(3)$ into the translation component of pose in $SE(3)$ through $\mathbf{r} = \mathbf{J}\rho$. This quantity appears in other situations as well when dealing with our matrix Lie groups, and we will learn later on in this chapter that this is called the (left) *Jacobian* of $SO(3)$. In this section, we will derive some alternate forms of this matrix that are sometimes useful.

We have defined \mathbf{J} as

$$\mathbf{J} = \sum_{n=0}^{\infty} \frac{1}{(n+1)!} (\phi^\wedge)^n. \quad (7.36)$$

By expanding this series and manipulating, we can show the following closed-form expressions for \mathbf{J} and its inverse:

$$\mathbf{J} = \frac{\sin \phi}{\phi} \mathbf{1} + \left(1 - \frac{\sin \phi}{\phi}\right) \mathbf{a} \mathbf{a}^T + \frac{1 - \cos \phi}{\phi} \mathbf{a}^\wedge, \quad (7.37a)$$

$$\mathbf{J}^{-1} = \frac{\phi}{2} \cot \frac{\phi}{2} \mathbf{1} + \left(1 - \frac{\phi}{2} \cot \frac{\phi}{2}\right) \mathbf{a} \mathbf{a}^T - \frac{\phi}{2} \mathbf{a}^\wedge, \quad (7.37b)$$

where $\phi = |\phi|$ is the angle of rotation and $\mathbf{a} = \phi/\phi$ is the unit-length axis of rotation. Owing to the nature of the $\cot(\phi/2)$ function, there are singularities associated with \mathbf{J} (i.e., the inverse does not exist) at $\phi = 2\pi m$ with m a non-zero integer.

Occasionally, we will come across the matrix $\mathbf{J}\mathbf{J}^T$ and its inverse. Starting with (7.37a), we can manipulate to show

$$\begin{aligned} \mathbf{J}\mathbf{J}^T &= \gamma \mathbf{1} + (1 - \gamma) \mathbf{a} \mathbf{a}^T, \quad (\mathbf{J}\mathbf{J}^T)^{-1} = \frac{1}{\gamma} \mathbf{1} + \left(1 - \frac{1}{\gamma}\right) \mathbf{a} \mathbf{a}^T, \\ \gamma &= 2 \frac{1 - \cos \phi}{\phi^2}. \end{aligned} \quad (7.38)$$

It turns out that $\mathbf{J}\mathbf{J}^T$ is positive-definite. There are two cases to consider, $\phi = 0$ and $\phi \neq 0$. For $\phi = 0$, we have $\mathbf{J}\mathbf{J}^T = \mathbf{1}$, which is positive-definite. For $\phi \neq 0$, we have for $\mathbf{x} \neq \mathbf{0}$ that

$$\begin{aligned}\mathbf{x}^T \mathbf{J}\mathbf{J}^T \mathbf{x} &= \mathbf{x}^T (\gamma\mathbf{1} + (1 - \gamma)\mathbf{a}\mathbf{a}^T) \mathbf{x} = \mathbf{x}^T (\mathbf{a}\mathbf{a}^T - \gamma\mathbf{a}^\wedge\mathbf{a}^\wedge) \mathbf{x} \\ &= \mathbf{x}^T \mathbf{a}\mathbf{a}^T \mathbf{x} + \gamma(\mathbf{a}^\wedge\mathbf{x})^T(\mathbf{a}^\wedge\mathbf{x}) \\ &= \underbrace{(\mathbf{a}^T \mathbf{x})^T(\mathbf{a}^T \mathbf{x})}_{\geq 0} + 2\underbrace{\frac{1 - \cos \phi}{\phi^2}}_{>0} \underbrace{(\mathbf{a}^\wedge\mathbf{x})^T(\mathbf{a}^\wedge\mathbf{x})}_{\geq 0} > 0, \quad (7.39)\end{aligned}$$

since the first term is only zero when \mathbf{a} and \mathbf{x} are perpendicular and the second term is only zero when \mathbf{x} and \mathbf{a} are parallel (these cannot happen at the same time). This shows that $\mathbf{J}\mathbf{J}^T$ is positive-definite.

It turns out that we can also write \mathbf{J} in terms of the rotation matrix, \mathbf{C} , associated with ϕ in the following way:

$$\mathbf{J} = \int_0^1 \mathbf{C}^\alpha d\alpha. \quad (7.40)$$

This can be seen through the following sequence of manipulations:

$$\begin{aligned}\int_0^1 \mathbf{C}^\alpha d\alpha &= \int_0^1 \exp(\phi^\wedge)^\alpha d\alpha = \int_0^1 \exp(\alpha \phi^\wedge) d\alpha \\ &= \int_0^1 \left(\sum_{n=0}^{\infty} \frac{1}{n!} \alpha^n (\phi^\wedge)^n \right) d\alpha = \sum_{n=0}^{\infty} \frac{1}{n!} \left(\int_0^1 \alpha^n d\alpha \right) (\phi^\wedge)^n \\ &= \sum_{n=0}^{\infty} \frac{1}{n!} \left(\frac{1}{n+1} \alpha^{n+1} \Big|_{\alpha=0}^{\alpha=1} \right) (\phi^\wedge)^n = \sum_{n=0}^{\infty} \frac{1}{(n+1)!} (\phi^\wedge)^n, \quad (7.41)\end{aligned}$$

which is the original series form of \mathbf{J} defined above.

Finally, we can also relate \mathbf{J} and \mathbf{C} through

$$\mathbf{C} = \mathbf{1} + \phi^\wedge \mathbf{J}, \quad (7.42)$$

but it is not possible to solve for \mathbf{J} in this expression since ϕ^\wedge is not invertible.

Direct Series Expression

We can also develop a direct series expression for \mathbf{T} from the exponential map by using the useful identity

$$(\boldsymbol{\xi}^\wedge)^4 + \phi^2 (\boldsymbol{\xi}^\wedge)^2 \equiv \mathbf{0}, \quad (7.43)$$

where $\boldsymbol{\xi} = \begin{bmatrix} \boldsymbol{\rho} \\ \phi \end{bmatrix}$ and $\phi = |\boldsymbol{\phi}|$. Expanding the series and using the identity to rewrite all terms quartic and higher in terms of lower-order terms,

we have

$$\begin{aligned}
\mathbf{T} &= \exp(\boldsymbol{\xi}^\wedge) \\
&= \sum_{n=0}^{\infty} \frac{1}{n!} (\boldsymbol{\xi}^\wedge)^n \\
&= \mathbf{1} + \boldsymbol{\xi}^\wedge + \frac{1}{2!}(\boldsymbol{\xi}^\wedge)^2 + \frac{1}{3!}(\boldsymbol{\xi}^\wedge)^3 + \frac{1}{4!}(\boldsymbol{\xi}^\wedge)^4 + \frac{1}{5!}(\boldsymbol{\xi}^\wedge)^5 + \dots \\
&= \mathbf{1} + \boldsymbol{\xi}^\wedge + \underbrace{\left(\frac{1}{2!} - \frac{1}{4!}\phi^2 + \frac{1}{6!}\phi^4 - \frac{1}{8!}\phi^6 + \dots \right)}_{\frac{1-\cos\phi}{\phi^2}} (\boldsymbol{\xi}^\wedge)^2 \\
&\quad + \underbrace{\left(\frac{1}{3!} - \frac{1}{5!}\phi^2 + \frac{1}{7!}\phi^4 - \frac{1}{9!}\phi^6 + \dots \right)}_{\frac{\phi-\sin\phi}{\phi^3}} (\boldsymbol{\xi}^\wedge)^3 \\
&= \mathbf{1} + \boldsymbol{\xi}^\wedge + \left(\frac{1-\cos\phi}{\phi^2} \right) (\boldsymbol{\xi}^\wedge)^2 + \left(\frac{\phi-\sin\phi}{\phi^3} \right) (\boldsymbol{\xi}^\wedge)^3. \tag{7.44}
\end{aligned}$$

Calculating \mathbf{T} using this approach avoids the need to deal with its constituent blocks.

7.1.4 Adjoints

A 6×6 transformation matrix, \mathcal{T} , can be constructed directly from the components of the 4×4 transformation matrix. We call this the *adjoint* of an element of $SE(3)$:

$$\mathcal{T} = \text{Ad}(\mathbf{T}) = \text{Ad} \left(\begin{bmatrix} \mathbf{C} & \mathbf{r} \\ \mathbf{0}^T & 1 \end{bmatrix} \right) = \begin{bmatrix} \mathbf{C} & \mathbf{r}^\wedge \mathbf{C} \\ \mathbf{0} & \mathbf{C} \end{bmatrix}. \tag{7.45}$$

We will abuse notation a bit and say that the set of adjoints of all the elements of $SE(3)$ is denoted

$$\text{Ad}(SE(3)) = \{\mathcal{T} = \text{Ad}(\mathbf{T}) \mid \mathbf{T} \in SE(3)\}. \tag{7.46}$$

It turns out that $\text{Ad}(SE(3))$ is also a matrix Lie group, which we show next.

Lie Group

For closure we let $\mathcal{T}_1 = \text{Ad}(\mathbf{T}_1)$, $\mathcal{T}_2 = \text{Ad}(\mathbf{T}_2) \in \text{Ad}(SE(3))$, and then

$$\begin{aligned}\mathcal{T}_1 \mathcal{T}_2 &= \text{Ad}(\mathbf{T}_1) \text{Ad}(\mathbf{T}_2) = \text{Ad} \left(\begin{bmatrix} \mathbf{C}_1 & \mathbf{r}_1 \\ \mathbf{0}^T & 1 \end{bmatrix} \right) \text{Ad} \left(\begin{bmatrix} \mathbf{C}_2 & \mathbf{r}_2 \\ \mathbf{0}^T & 1 \end{bmatrix} \right) \\ &= \begin{bmatrix} \mathbf{C}_1 & \mathbf{r}_1^\wedge \mathbf{C}_1 \\ \mathbf{0} & \mathbf{C}_1 \end{bmatrix} \begin{bmatrix} \mathbf{C}_2 & \mathbf{r}_2^\wedge \mathbf{C}_2 \\ \mathbf{0} & \mathbf{C}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{C}_1 \mathbf{C}_2 & \mathbf{C}_1 \mathbf{r}_2^\wedge \mathbf{C}_2 + \mathbf{r}_1^\wedge \mathbf{C}_1 \mathbf{C}_2 \\ \mathbf{0} & \mathbf{C}_1 \mathbf{C}_2 \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{C}_1 \mathbf{C}_2 & (\mathbf{C}_1 \mathbf{r}_2 + \mathbf{r}_1)^\wedge \mathbf{C}_1 \mathbf{C}_2 \\ \mathbf{0} & \mathbf{C}_1 \mathbf{C}_2 \end{bmatrix} \\ &= \text{Ad} \left(\begin{bmatrix} \mathbf{C}_1 \mathbf{C}_2 & \mathbf{C}_1 \mathbf{r}_2 + \mathbf{r}_1 \\ \mathbf{0}^T & 1 \end{bmatrix} \right) \in \text{Ad}(SE(3)),\end{aligned}\quad (7.47)$$

where we have used the nice property that

$$\mathbf{C} \mathbf{v}^\wedge \mathbf{C}^T = (\mathbf{C} \mathbf{v})^\wedge,\quad (7.48)$$

for any $\mathbf{C} \in SO(3)$ and $\mathbf{v} \in \mathbb{R}^3$. Associativity follows from basic properties of matrix multiplication, and the identity element of the group is the 6×6 identity matrix. For invertibility, we let $\mathcal{T} = \text{Ad}(\mathbf{T}) \in \text{Ad}(SE(3))$, and then we have

$$\begin{aligned}\mathcal{T}^{-1} &= \text{Ad}(\mathbf{T})^{-1} = \text{Ad} \left(\begin{bmatrix} \mathbf{C} & \mathbf{r} \\ \mathbf{0}^T & 1 \end{bmatrix} \right)^{-1} = \begin{bmatrix} \mathbf{C} & \mathbf{r}^\wedge \mathbf{C} \\ \mathbf{0} & \mathbf{C} \end{bmatrix}^{-1} \\ &= \begin{bmatrix} \mathbf{C}^T & -\mathbf{C}^T \mathbf{r}^\wedge \\ \mathbf{0} & \mathbf{C}^T \end{bmatrix} = \begin{bmatrix} \mathbf{C}^T & (-\mathbf{C}^T \mathbf{r})^\wedge \mathbf{C}^T \\ \mathbf{0} & \mathbf{C}^T \end{bmatrix} \\ &= \text{Ad} \left(\begin{bmatrix} \mathbf{C}^T & -\mathbf{C}^T \mathbf{r} \\ \mathbf{0}^T & 1 \end{bmatrix} \right) = \text{Ad}(\mathbf{T}^{-1}) \in \text{Ad}(SE(3)).\end{aligned}\quad (7.49)$$

Other than smoothness, these four properties show that $\text{Ad}(SE(3))$ is a matrix Lie group.

Lie Algebra

We can also talk about the *adjoint* of an element of $\mathfrak{se}(3)$. Let $\Xi = \xi^\wedge \in \mathfrak{se}(3)$; then the adjoint of this element is

$$\text{ad}(\Xi) = \text{ad}(\xi^\wedge) = \xi^\wedge,\quad (7.50)$$

where

$$\xi^\wedge = \begin{bmatrix} \rho \\ \phi \end{bmatrix}^\wedge = \begin{bmatrix} \phi^\wedge & \rho^\wedge \\ \mathbf{0} & \phi^\wedge \end{bmatrix} \in \mathbb{R}^{6 \times 6}, \quad \rho, \phi \in \mathbb{R}^3.\quad (7.51)$$

Note that we have used uppercase, $\text{Ad}(\cdot)$, for the adjoint of $SE(3)$ and lowercase, $\text{ad}(\cdot)$, for the adjoint of $\mathfrak{se}(3)$.

The Lie algebra associated with $\text{Ad}(SE(3))$ is given by

vectorspace: $\text{ad}(\mathfrak{se}(3)) = \{\Psi = \text{ad}(\Xi) \in \mathbb{R}^{6 \times 6} \mid \Xi \in \mathfrak{se}(3)\}$,

field: \mathbb{R} ,

Lie bracket: $[\Psi_1, \Psi_2] = \Psi_1 \Psi_2 - \Psi_2 \Psi_1$.

Again, we will omit showing that $\text{ad}(\mathfrak{se}(3))$ is a vectorspace, but will briefly show that the four Lie bracket properties hold. Let $\Psi, \Psi_1 = \xi_1^\wedge, \Psi_2 = \xi_2^\wedge \in \text{ad}(\mathfrak{se}(3))$. Then, for the closure property, we have

$$[\Psi_1, \Psi_2] = \Psi_1 \Psi_2 - \Psi_2 \Psi_1 = \xi_1^\wedge \xi_2^\wedge - \xi_2^\wedge \xi_1^\wedge = \underbrace{(\xi_1^\wedge \xi_2)}_{\in \mathbb{R}^6}^\wedge \in \text{ad}(\mathfrak{se}(3)). \quad (7.52)$$

Bilinearity follows directly from the fact that $(\cdot)^\wedge$ is a linear operator. The alternating property can be seen easily through

$$[\Psi, \Psi] = \Psi \Psi - \Psi \Psi = \mathbf{0} \in \text{ad}(\mathfrak{se}(3)). \quad (7.53)$$

Finally, the Jacobi identity can be verified by substituting and applying the definition of the Lie bracket. Again, we will refer to $\text{ad}(\mathfrak{se}(3))$ as the Lie algebra, although technically this is only the associated vectorspace.

Exponential Map

Another issue to discuss is the relationship between $\text{Ad}(SE(3))$ and $\text{ad}(\mathfrak{se}(3))$ through the exponential map. Not surprisingly, we have that

$$\mathcal{T} = \exp(\xi^\wedge) = \sum_{n=0}^{\infty} \frac{1}{n!} (\xi^\wedge)^n, \quad (7.54)$$

where $\mathcal{T} \in \text{Ad}(SE(3))$ and $\xi \in \mathbb{R}^6$ (and hence $\xi^\wedge \in \text{ad}(\mathfrak{se}(3))$). We can go in the other direction using

$$\xi = \ln(\mathcal{T})^\vee, \quad (7.55)$$

where \vee undoes the \wedge operation. The exponential mapping is again surjective-only, which we discuss below.

First, however, we note that here is a nice commutative relationship between the various Lie groups and algebras associated with poses:

$$\begin{array}{ccc} & \text{Lie algebra} & \text{Lie group} \\ 4 \times 4 & \xi^\wedge \in \mathfrak{se}(3) & \xrightarrow{\exp} \mathbf{T} \in SE(3) \\ & \downarrow \text{ad} & \downarrow \text{Ad} \\ 6 \times 6 & \xi^\wedge \in \text{ad}(\mathfrak{se}(3)) & \xrightarrow{\exp} \mathcal{T} \in \text{Ad}(SE(3)) \end{array} \quad (7.56)$$

We could draw on this commutative relationship to claim the surjective-only property of the exponential map from $\text{ad}(\mathfrak{se}(3))$ to $\text{Ad}(SE(3))$ by going the long way around the loop, since we have already shown that this path exists. However, it is also possible to show it directly, which

amounts to showing that

$$\underbrace{\text{Ad}(\exp(\xi^\wedge))}_{\mathcal{T}} = \exp\left(\underbrace{\text{ad}(\xi^\wedge)}_{\xi^\wedge}\right), \quad (7.57)$$

since this implies that we can go from $\xi \in \mathbb{R}^6$ to $\mathcal{T} \in \text{Ad}(SE(3))$ and back.

To see this, let $\xi = [\rho]$, and then starting from the right-hand side, we have

$$\begin{aligned} \exp(\text{ad}(\xi^\wedge)) &= \exp(\xi^\wedge) = \sum_{n=0}^{\infty} \frac{1}{n!} (\xi^\wedge)^n \\ &= \sum_{n=0}^{\infty} \frac{1}{n!} \begin{bmatrix} \phi^\wedge & \rho^\wedge \\ \mathbf{0} & \phi^\wedge \end{bmatrix}^n = \begin{bmatrix} \mathbf{C} & \mathbf{K} \\ \mathbf{0} & \mathbf{C} \end{bmatrix}, \end{aligned} \quad (7.58)$$

where \mathbf{C} is the usual expression for the rotation matrix in terms of ϕ and

$$\mathbf{K} = \sum_{n=0}^{\infty} \sum_{m=0}^{\infty} \frac{1}{(n+m+1)!} (\phi^\wedge)^n \rho^\wedge (\phi^\wedge)^m,$$

which can be found through careful manipulation. Starting from the left-hand side, we have

$$\text{Ad}(\exp(\xi^\wedge)) = \text{Ad}\left(\begin{bmatrix} \mathbf{C} & \mathbf{J}\rho \\ \mathbf{0}^T & 1 \end{bmatrix}\right) = \begin{bmatrix} \mathbf{C} & (\mathbf{J}\rho)^\wedge \mathbf{C} \\ \mathbf{0} & \mathbf{C} \end{bmatrix}, \quad (7.59)$$

where \mathbf{J} is given in (7.36). Comparing (7.58) and (7.59), what remains to be shown is the equivalence of the top-right block: $\mathbf{K} = (\mathbf{J}\rho)^\wedge \mathbf{C}$. To see this, we use the following sequence of manipulations:

$$\begin{aligned} (\mathbf{J}\rho)^\wedge \mathbf{C} &= \left(\int_0^1 \mathbf{C}^\alpha d\alpha \rho\right)^\wedge \mathbf{C} = \int_0^1 (\mathbf{C}^\alpha \rho)^\wedge \mathbf{C} d\alpha \\ &= \int_0^1 \mathbf{C}^\alpha \rho^\wedge \mathbf{C}^{1-\alpha} d\alpha = \int_0^1 \exp(\alpha \phi^\wedge) \rho^\wedge \exp((1-\alpha)\phi^\wedge) d\alpha \\ &= \int_0^1 \left(\sum_{n=0}^{\infty} \frac{1}{n!} (\alpha \phi^\wedge)^n\right) \rho^\wedge \left(\sum_{m=0}^{\infty} \frac{1}{m!} ((1-\alpha)\phi^\wedge)^m\right) d\alpha \\ &= \sum_{n=0}^{\infty} \sum_{m=0}^{\infty} \frac{1}{n! m!} \left(\int_0^1 \alpha^n (1-\alpha)^m d\alpha\right) (\phi^\wedge)^n \rho^\wedge (\phi^\wedge)^m, \end{aligned} \quad (7.60)$$

where we have used that \wedge is linear and that $(\mathbf{C}\mathbf{v})^\wedge = \mathbf{C}\mathbf{v}^\wedge \mathbf{C}^T$. After several integrations by parts we can show that

$$\int_0^1 \alpha^n (1-\alpha)^m d\alpha = \frac{n! m!}{(n+m+1)!}, \quad (7.61)$$

and therefore, $\mathbf{K} = (\mathbf{J}\rho)^\wedge \mathbf{C}$, which is the desired result.

Direct Series Expression

Similarly to the direct series expression for \mathbf{T} , we can also work one out for $\mathcal{T} = \text{Ad}(\mathbf{T})$ by using the identity

$$(\boldsymbol{\xi}^\wedge)^5 + 2\phi^2 (\boldsymbol{\xi}^\wedge)^3 + \phi^4 \boldsymbol{\xi}^\wedge \equiv \mathbf{0}, \quad (7.62)$$

where $\boldsymbol{\xi} = \begin{bmatrix} \rho \\ \phi \end{bmatrix}$ and $\phi = |\phi|$. Expanding the series and using the identity to rewrite all terms quintic and higher in lower-order terms, we have

$$\begin{aligned} \mathcal{T} &= \exp(\boldsymbol{\xi}^\wedge) \\ &= \sum_{n=0}^{\infty} \frac{1}{n!} (\boldsymbol{\xi}^\wedge)^n \\ &= \mathbf{1} + \boldsymbol{\xi}^\wedge + \frac{1}{2!} (\boldsymbol{\xi}^\wedge)^2 + \frac{1}{3!} (\boldsymbol{\xi}^\wedge)^3 + \frac{1}{4!} (\boldsymbol{\xi}^\wedge)^4 + \frac{1}{5!} (\boldsymbol{\xi}^\wedge)^5 + \dots \\ &= \mathbf{1} + \underbrace{\left(1 - \frac{1}{5!}\phi^4 + \frac{2}{7!}\phi^6 - \frac{3}{9!}\phi^8 + \frac{4}{11!}\phi^{10} - \dots\right)}_{\frac{3\sin\phi - \phi\cos\phi}{2\phi}} \boldsymbol{\xi}^\wedge \\ &\quad + \underbrace{\left(\frac{1}{2!} - \frac{1}{6!}\phi^4 + \frac{2}{8!}\phi^6 - \frac{3}{10!}\phi^8 + \frac{4}{12!}\phi^{10} - \dots\right)}_{\frac{4 - \phi\sin\phi - 4\cos\phi}{2\phi^2}} (\boldsymbol{\xi}^\wedge)^2 \\ &\quad + \underbrace{\left(\frac{1}{3!} - \frac{2}{5!}\phi^2 + \frac{3}{7!}\phi^4 - \frac{4}{9!}\phi^6 + \frac{5}{11!}\phi^8 - \dots\right)}_{\frac{\sin\phi - \phi\cos\phi}{2\phi^3}} (\boldsymbol{\xi}^\wedge)^3 \\ &\quad + \underbrace{\left(\frac{1}{4!} - \frac{2}{6!}\phi^2 + \frac{3}{8!}\phi^4 - \frac{4}{10!}\phi^6 + \frac{5}{12!}\phi^8 - \dots\right)}_{\frac{2 - \phi\sin\phi - 2\cos\phi}{2\phi^4}} (\boldsymbol{\xi}^\wedge)^4 \\ &= \mathbf{1} + \left(\frac{3\sin\phi - \phi\cos\phi}{2\phi}\right) \boldsymbol{\xi}^\wedge + \left(\frac{4 - \phi\sin\phi - 4\cos\phi}{2\phi^2}\right) (\boldsymbol{\xi}^\wedge)^2 \\ &\quad + \left(\frac{\sin\phi - \phi\cos\phi}{2\phi^3}\right) (\boldsymbol{\xi}^\wedge)^3 + \left(\frac{2 - \phi\sin\phi - 2\cos\phi}{2\phi^4}\right) (\boldsymbol{\xi}^\wedge)^4. \end{aligned} \quad (7.63)$$

As in the 4×4 case, this last expression allows us to evaluate \mathcal{T} without working with its constituent blocks.

7.1.5 Baker-Campbell-Hausdorff

We can combine two scalar exponential functions as follows:

$$\exp(a) \exp(b) = \exp(a + b), \quad (7.64)$$

where $a, b \in \mathbb{R}$. Unfortunately, this is not so easy for the matrix case. To compound two matrix exponentials, we use the *Baker-Campbell-Hausdorff (BCH)* formula:

$$\begin{aligned} & \ln(\exp(\mathbf{A})\exp(\mathbf{B})) \\ &= \sum_{n=1}^{\infty} \frac{(-1)^{n-1}}{n} \sum_{\substack{r_i + s_i > 0, \\ 1 \leq i \leq n}} \frac{(\sum_{i=1}^n (r_i + s_i))^{-1}}{\prod_{i=1}^n r_i! s_i!} [\mathbf{A}^{r_1} \mathbf{B}^{s_1} \mathbf{A}^{r_2} \mathbf{B}^{s_2} \cdots \mathbf{A}^{r_n} \mathbf{B}^{s_n}], \end{aligned} \quad (7.65)$$

where

$$\begin{aligned} & [\mathbf{A}^{r_1} \mathbf{B}^{s_1} \mathbf{A}^{r_2} \mathbf{B}^{s_2} \cdots \mathbf{A}^{r_n} \mathbf{B}^{s_n}] \\ &= \underbrace{[\mathbf{A}, \dots, [\mathbf{A},}_{r_1} \underbrace{[\mathbf{B}, \dots, [\mathbf{B},}_{s_1} \underbrace{\dots, [\mathbf{A}, \dots, [\mathbf{A},}_{r_n} \underbrace{[\mathbf{B}, \dots, [\mathbf{B}, \mathbf{B}] \dots] \dots] \dots] \dots], \end{aligned} \quad (7.66)$$

which is zero if $s_n > 1$ or if $s_n = 0$ and $r_n > 1$. The Lie bracket is the usual

$$[\mathbf{A}, \mathbf{B}] = \mathbf{AB} - \mathbf{BA}. \quad (7.67)$$

Note that the BCH formula is an infinite series. In the event that $[\mathbf{A}, \mathbf{B}] = \mathbf{0}$, the BCH formula simplifies to

$$\ln(\exp(\mathbf{A})\exp(\mathbf{B})) = \mathbf{A} + \mathbf{B}, \quad (7.68)$$

but this case is not particularly useful to us, except as an approximation. The first several terms of the general BCH formula are

$$\begin{aligned} & \ln(\exp(\mathbf{A})\exp(\mathbf{B})) = \mathbf{A} + \mathbf{B} + \frac{1}{2} [\mathbf{A}, \mathbf{B}] \\ &+ \frac{1}{12} [\mathbf{A}, [\mathbf{A}, \mathbf{B}]] - \frac{1}{12} [\mathbf{B}, [\mathbf{A}, \mathbf{B}]] - \frac{1}{24} [\mathbf{B}, [\mathbf{A}, [\mathbf{A}, \mathbf{B}]]] \\ &- \frac{1}{720} ([[[[\mathbf{A}, \mathbf{B}], \mathbf{B}], \mathbf{B}], \mathbf{B}] + [[[[\mathbf{B}, \mathbf{A}], \mathbf{A}], \mathbf{A}], \mathbf{A}]) \\ &+ \frac{1}{360} ([[[[\mathbf{A}, \mathbf{B}], \mathbf{B}], \mathbf{B}], \mathbf{A}] + [[[[\mathbf{B}, \mathbf{A}], \mathbf{A}], \mathbf{A}], \mathbf{B}]) \\ &+ \frac{1}{120} ([[[[\mathbf{A}, \mathbf{B}], \mathbf{A}], \mathbf{B}], \mathbf{A}] + [[[[\mathbf{B}, \mathbf{A}], \mathbf{B}], \mathbf{A}], \mathbf{B}]) + \cdots. \end{aligned} \quad (7.69)$$

If we keep only terms linear in \mathbf{A} , the general BCH formula becomes (Klarsfeld and Oteo, 1989)

$$\ln(\exp(\mathbf{A})\exp(\mathbf{B})) \approx \mathbf{B} + \sum_{n=0}^{\infty} \frac{B_n}{n!} \underbrace{[\mathbf{B}, [\mathbf{B}, \dots, [\mathbf{B}, \mathbf{A}] \dots]]}_n. \quad (7.70)$$

Henry Frederick Baker (1866-1956) was a British mathematician, working mainly in algebraic geometry, but also remembered for contributions to partial differential equations and Lie groups. John Edward Campbell (1862-1924) was a British mathematician, best known for his contribution to the BCH formula and a 1903 book popularizing the ideas of Sophus Lie. Felix Hausdorff (1868-1942) was a German mathematician who is considered to be one of the founders of modern topology and who contributed significantly to set theory, descriptive set theory, measure theory, function theory, and functional analysis. Henri Poincaré (1854-1912) is also said to have had a hand in the BCH formula.

If we keep only terms linear in \mathbf{B} , the general BCH formula becomes

$$\ln(\exp(\mathbf{A})\exp(\mathbf{B})) \approx \mathbf{A} + \sum_{n=0}^{\infty} (-1)^n \frac{B_n}{n!} \underbrace{[\mathbf{A}, [\mathbf{A}, \dots [\mathbf{A}, \mathbf{B}] \dots]]}_n. \quad (7.71)$$

The Bernoulli numbers were discovered around the same time by the Swiss mathematician Jakob Bernoulli (1655-1705), after whom they are named, and independently by Japanese mathematician Seki Kōwa (1642-1708). Seki's discovery was posthumously published in 1712 in his work *Katsuyo Sampo*; Bernoulli's, also posthumously, in his *Ars Conjectandi (The Art of Conjecture)* of 1713. Ada Lovelace's *Note G* on the analytical engine from 1842 describes an algorithm for generating Bernoulli numbers with Babbage's machine. As a result, the Bernoulli numbers have the distinction of being the subject of the first computer program.

The B_n are the *Bernoulli numbers*¹¹,

$$\begin{aligned} B_0 &= 1, B_1 = -\frac{1}{2}, B_2 = \frac{1}{6}, B_3 = 0, B_4 = -\frac{1}{30}, B_5 = 0, B_6 = \frac{1}{42}, \\ B_7 &= 0, B_8 = -\frac{1}{30}, B_9 = 0, B_{10} = \frac{5}{66}, B_{11} = 0, B_{12} = -\frac{691}{2730}, \\ B_{13} &= 0, B_{14} = \frac{7}{6}, B_{15} = 0, \dots, \end{aligned} \quad (7.72)$$

which appear frequently in number theory. It is also worth noting that $B_n = 0$ for all odd $n > 1$, which reduces the number of terms that need to be implemented in approximations of some of our infinite series.

The *Lie product formula*,

$$\exp(\mathbf{A} + \mathbf{B}) = \lim_{\alpha \rightarrow \infty} (\exp(\mathbf{A}/\alpha)\exp(\mathbf{B}/\alpha))^{\alpha}, \quad (7.73)$$

provides another way of looking at compounding matrix exponentials; compounding is effectively slicing each matrix exponential into an infinite number of infinitely thin slices and then interleaving the slices. We next discuss application of the general BCH formula to the specific cases of rotations and poses.

Rotations

In the particular case of $SO(3)$, we can show that

$$\begin{aligned} \ln(\mathbf{C}_1 \mathbf{C}_2)^\vee &= \ln(\exp(\phi_1^\wedge) \exp(\phi_2^\wedge))^\vee \\ &= \phi_1 + \phi_2 + \frac{1}{2}\phi_1^\wedge \phi_2 + \frac{1}{12}\phi_1^\wedge \phi_1^\wedge \phi_2 + \frac{1}{12}\phi_2^\wedge \phi_2^\wedge \phi_1 + \dots, \end{aligned} \quad (7.74)$$

where $\mathbf{C}_1 = \exp(\phi_1^\wedge)$, $\mathbf{C}_2 = \exp(\phi_2^\wedge) \in SO(3)$. Alternatively, if we assume that ϕ_1 or ϕ_2 is small, then using the approximate BCH formulas we can show that

$$\begin{aligned} \ln(\mathbf{C}_1 \mathbf{C}_2)^\vee &= \ln(\exp(\phi_1^\wedge) \exp(\phi_2^\wedge))^\vee \\ &\approx \begin{cases} \mathbf{J}_\ell(\phi_2)^{-1} \phi_1 + \phi_2 & \text{if } \phi_1 \text{ small} \\ \phi_1 + \mathbf{J}_r(\phi_1)^{-1} \phi_2 & \text{if } \phi_2 \text{ small} \end{cases}, \end{aligned} \quad (7.75)$$

¹¹ Technically, the sequence shown is the *first* Bernoulli sequence. There is also a *second* sequence in which $B_1 = \frac{1}{2}$, but we will not need it here.

where

$$\mathbf{J}_r(\phi)^{-1} = \sum_{n=0}^{\infty} \frac{B_n}{n!} (-\phi^\wedge)^n = \frac{\phi}{2} \cot \frac{\phi}{2} \mathbf{1} + \left(1 - \frac{\phi}{2} \cot \frac{\phi}{2}\right) \mathbf{a} \mathbf{a}^T + \frac{\phi}{2} \mathbf{a}^\wedge, \quad (7.76a)$$

$$\mathbf{J}_\ell(\phi)^{-1} = \sum_{n=0}^{\infty} \frac{B_n}{n!} (\phi^\wedge)^n = \frac{\phi}{2} \cot \frac{\phi}{2} \mathbf{1} + \left(1 - \frac{\phi}{2} \cot \frac{\phi}{2}\right) \mathbf{a} \mathbf{a}^T - \frac{\phi}{2} \mathbf{a}^\wedge. \quad (7.76b)$$

In Lie group theory, \mathbf{J}_r and \mathbf{J}_ℓ are referred to as the *right* and *left Jacobians* of $SO(3)$, respectively. As noted earlier, due to the nature of the $\cot(\phi/2)$ function, there are singularities associated with $\mathbf{J}_r, \mathbf{J}_\ell$ at $\phi = 2\pi m$ with m a non-zero integer. Inverting, we have the following expressions for the Jacobians:

$$\begin{aligned} \mathbf{J}_r(\phi) &= \sum_{n=0}^{\infty} \frac{1}{(n+1)!} (-\phi^\wedge)^n = \int_0^1 \mathbf{C}^{-\alpha} d\alpha \\ &= \frac{\sin \phi}{\phi} \mathbf{1} + \left(1 - \frac{\sin \phi}{\phi}\right) \mathbf{a} \mathbf{a}^T - \frac{1 - \cos \phi}{\phi} \mathbf{a}^\wedge, \end{aligned} \quad (7.77a)$$

$$\begin{aligned} \mathbf{J}_\ell(\phi) &= \sum_{n=0}^{\infty} \frac{1}{(n+1)!} (\phi^\wedge)^n = \int_0^1 \mathbf{C}^\alpha d\alpha \\ &= \frac{\sin \phi}{\phi} \mathbf{1} + \left(1 - \frac{\sin \phi}{\phi}\right) \mathbf{a} \mathbf{a}^T + \frac{1 - \cos \phi}{\phi} \mathbf{a}^\wedge, \end{aligned} \quad (7.77b)$$

where $\mathbf{C} = \exp(\phi^\wedge)$, $\phi = |\phi|$, and $\mathbf{a} = \phi/\phi$. We draw attention to the fact that

$$\mathbf{J}_\ell(\phi) = \mathbf{C} \mathbf{J}_r(\phi), \quad (7.78)$$

which allows us to relate one Jacobian to the other. To show this is fairly straightforward, using the definitions:

$$\begin{aligned} \mathbf{C} \mathbf{J}_r(\phi) &= \mathbf{C} \int_0^1 \mathbf{C}^{-\alpha} d\alpha = \int_0^1 \mathbf{C}^{1-\alpha} d\alpha \\ &= - \int_1^0 \mathbf{C}^\beta d\beta = \int_0^1 \mathbf{C}^\beta d\beta = \mathbf{J}_\ell(\phi). \end{aligned} \quad (7.79)$$

Another relationship between the left and right Jacobians is:

$$\mathbf{J}_\ell(-\phi) = \mathbf{J}_r(\phi), \quad (7.80)$$

which is again fairly easy to see:

$$\begin{aligned} \mathbf{J}_r(\phi) &= \int_0^1 \mathbf{C}(\phi)^{-\alpha} d\alpha = \int_0^1 (\mathbf{C}(\phi)^{-1})^\alpha d\alpha \\ &= \int_0^1 (\mathbf{C}(-\phi))^\alpha d\alpha = \mathbf{J}_\ell(-\phi). \end{aligned} \quad (7.81)$$

We next look at $SE(3)$.

Poses

In the particular cases of $SE(3)$ and $\text{Ad}(SE(3))$, we can show that

$$\begin{aligned} \ln(\mathbf{T}_1 \mathbf{T}_2)^\vee &= \ln(\exp(\boldsymbol{\xi}_1^\wedge) \exp(\boldsymbol{\xi}_2^\wedge))^\vee \\ &= \boldsymbol{\xi}_1 + \boldsymbol{\xi}_2 + \frac{1}{2}\boldsymbol{\xi}_1^\wedge \boldsymbol{\xi}_2 + \frac{1}{12}\boldsymbol{\xi}_1^\wedge \boldsymbol{\xi}_1^\wedge \boldsymbol{\xi}_2 + \frac{1}{12}\boldsymbol{\xi}_2^\wedge \boldsymbol{\xi}_2^\wedge \boldsymbol{\xi}_1 + \dots, \end{aligned} \quad (7.82\text{a})$$

$$\begin{aligned} \ln(\mathcal{T}_1 \mathcal{T}_2)^\vee &= \ln(\exp(\boldsymbol{\xi}_1^\wedge) \exp(\boldsymbol{\xi}_2^\wedge))^\vee \\ &= \boldsymbol{\xi}_1 + \boldsymbol{\xi}_2 + \frac{1}{2}\boldsymbol{\xi}_1^\wedge \boldsymbol{\xi}_2 + \frac{1}{12}\boldsymbol{\xi}_1^\wedge \boldsymbol{\xi}_1^\wedge \boldsymbol{\xi}_2 + \frac{1}{12}\boldsymbol{\xi}_2^\wedge \boldsymbol{\xi}_2^\wedge \boldsymbol{\xi}_1 + \dots, \end{aligned} \quad (7.82\text{b})$$

where $\mathbf{T}_1 = \exp(\boldsymbol{\xi}_1^\wedge)$, $\mathbf{T}_2 = \exp(\boldsymbol{\xi}_2^\wedge) \in SE(3)$, and $\mathcal{T}_1 = \exp(\boldsymbol{\xi}_1^\wedge)$, $\mathcal{T}_2 = \exp(\boldsymbol{\xi}_2^\wedge) \in \text{Ad}(SE(3))$. Alternatively, if we assume that $\boldsymbol{\xi}_1$ or $\boldsymbol{\xi}_2$ is small, then using the approximate BCH formulas, we can show that

$$\begin{aligned} \ln(\mathbf{T}_1 \mathbf{T}_2)^\vee &= \ln(\exp(\boldsymbol{\xi}_1^\wedge) \exp(\boldsymbol{\xi}_2^\wedge))^\vee \\ &\approx \begin{cases} \mathcal{J}_\ell(\boldsymbol{\xi}_2)^{-1} \boldsymbol{\xi}_1 + \boldsymbol{\xi}_2 & \text{if } \boldsymbol{\xi}_1 \text{ small} \\ \boldsymbol{\xi}_1 + \mathcal{J}_r(\boldsymbol{\xi}_1)^{-1} \boldsymbol{\xi}_2 & \text{if } \boldsymbol{\xi}_2 \text{ small} \end{cases}, \end{aligned} \quad (7.83\text{a})$$

$$\begin{aligned} \ln(\mathcal{T}_1 \mathcal{T}_2)^\vee &= \ln(\exp(\boldsymbol{\xi}_1^\wedge) \exp(\boldsymbol{\xi}_2^\wedge))^\vee \\ &\approx \begin{cases} \mathcal{J}_\ell(\boldsymbol{\xi}_2)^{-1} \boldsymbol{\xi}_1 + \boldsymbol{\xi}_2 & \text{if } \boldsymbol{\xi}_1 \text{ small} \\ \boldsymbol{\xi}_1 + \mathcal{J}_r(\boldsymbol{\xi}_1)^{-1} \boldsymbol{\xi}_2 & \text{if } \boldsymbol{\xi}_2 \text{ small} \end{cases}, \end{aligned} \quad (7.83\text{b})$$

where

$$\mathcal{J}_r(\boldsymbol{\xi})^{-1} = \sum_{n=0}^{\infty} \frac{B_n}{n!} (-\boldsymbol{\xi}^\wedge)^n, \quad (7.84\text{a})$$

$$\mathcal{J}_\ell(\boldsymbol{\xi})^{-1} = \sum_{n=0}^{\infty} \frac{B_n}{n!} (\boldsymbol{\xi}^\wedge)^n. \quad (7.84\text{b})$$

In Lie group theory, \mathcal{J}_r and \mathcal{J}_ℓ are referred to as the *right* and *left Jacobians* of $SE(3)$, respectively. Inverting, we have the following expressions for the Jacobians:

$$\mathcal{J}_r(\boldsymbol{\xi}) = \sum_{n=0}^{\infty} \frac{1}{(n+1)!} (-\boldsymbol{\xi}^\wedge)^n = \int_0^1 \mathcal{T}^{-\alpha} d\alpha = \begin{bmatrix} \mathbf{J}_r & \mathbf{Q}_r \\ \mathbf{0} & \mathbf{J}_r \end{bmatrix}, \quad (7.85\text{a})$$

$$\mathcal{J}_\ell(\boldsymbol{\xi}) = \sum_{n=0}^{\infty} \frac{1}{(n+1)!} (\boldsymbol{\xi}^\wedge)^n = \int_0^1 \mathcal{T}^\alpha d\alpha = \begin{bmatrix} \mathbf{J}_\ell & \mathbf{Q}_\ell \\ \mathbf{0} & \mathbf{J}_\ell \end{bmatrix}, \quad (7.85\text{b})$$

where

$$\mathbf{Q}_\ell(\boldsymbol{\xi}) = \sum_{n=0}^{\infty} \sum_{m=0}^{\infty} \frac{1}{(n+m+2)!} (\phi^\wedge)^n \boldsymbol{\rho}^\wedge (\phi^\wedge)^m \quad (7.86a)$$

$$\begin{aligned} &= \frac{1}{2} \boldsymbol{\rho}^\wedge + \left(\frac{\phi - \sin \phi}{\phi^3} \right) (\phi^\wedge \boldsymbol{\rho}^\wedge + \boldsymbol{\rho}^\wedge \phi^\wedge + \phi^\wedge \boldsymbol{\rho}^\wedge \phi^\wedge) \\ &\quad + \left(\frac{\phi^2 + 2 \cos \phi - 2}{2\phi^4} \right) (\phi^\wedge \phi^\wedge \boldsymbol{\rho}^\wedge + \boldsymbol{\rho}^\wedge \phi^\wedge \phi^\wedge - 3\phi^\wedge \boldsymbol{\rho}^\wedge \phi^\wedge) \\ &\quad + \left(\frac{2\phi - 3 \sin \phi + \phi \cos \phi}{2\phi^5} \right) (\phi^\wedge \boldsymbol{\rho}^\wedge \phi^\wedge \phi^\wedge + \phi^\wedge \phi^\wedge \boldsymbol{\rho}^\wedge \phi^\wedge), \end{aligned} \quad (7.86b)$$

$$\mathbf{Q}_r(\boldsymbol{\xi}) = \mathbf{Q}_\ell(-\boldsymbol{\xi}) = \mathbf{C} \mathbf{Q}_\ell(\boldsymbol{\xi}) + (\mathbf{J}_\ell \boldsymbol{\rho})^\wedge \mathbf{C} \mathbf{J}_\ell, \quad (7.86c)$$

and $\mathcal{T} = \exp(\boldsymbol{\xi}^\wedge)$, $\mathbf{T} = \exp(\boldsymbol{\xi}^\wedge)$, $\mathbf{C} = \exp(\phi^\wedge)$, $\boldsymbol{\xi} = \begin{bmatrix} \boldsymbol{\rho} \\ \phi \end{bmatrix}$. The expression for \mathbf{Q}_ℓ comes from expanding the series and grouping terms into the series forms of the trigonometric functions¹². The relations for \mathbf{Q}_r come from the relationships between the left and right Jacobians:

$$\mathcal{J}_\ell(\boldsymbol{\xi}) = \mathcal{T} \mathcal{J}_r(\boldsymbol{\xi}), \quad \mathcal{J}_\ell(-\boldsymbol{\xi}) = \mathcal{J}_r(\boldsymbol{\xi}). \quad (7.87)$$

The first can be seen to be true from

$$\begin{aligned} \mathcal{T} \mathcal{J}_r(\boldsymbol{\xi}) &= \mathcal{T} \int_0^1 \mathcal{T}^{-\alpha} d\alpha = \int_0^1 \mathcal{T}^{1-\alpha} d\alpha \\ &= - \int_1^0 \mathcal{T}^\beta d\beta = \int_0^1 \mathcal{T}^\beta d\beta = \mathcal{J}_\ell(\boldsymbol{\xi}), \end{aligned} \quad (7.88)$$

and the second from

$$\begin{aligned} \mathcal{J}_r(\boldsymbol{\xi}) &= \int_0^1 \mathcal{T}(\boldsymbol{\xi})^{-\alpha} d\alpha = \int_0^1 (\mathcal{T}(\boldsymbol{\xi})^{-1})^\alpha d\alpha \\ &= \int_0^1 (\mathcal{T}(-\boldsymbol{\xi}))^\alpha d\alpha = \mathcal{J}_\ell(-\boldsymbol{\xi}). \end{aligned} \quad (7.89)$$

We can also work out a direct series expression for \mathcal{J}_ℓ using the results of Section 7.1.4. From the form of the series expressions, we have that

$$\mathcal{T} \equiv \mathbf{1} + \boldsymbol{\xi}^\wedge \mathcal{J}_\ell. \quad (7.90)$$

Expanding the expression for the Jacobian, we see that

$$\mathcal{J}_\ell = \sum_{n=0}^{\infty} \frac{1}{(n+1)!} (\boldsymbol{\xi}^\wedge)^n = \mathbf{1} + \alpha_1 \boldsymbol{\xi}^\wedge + \alpha_2 (\boldsymbol{\xi}^\wedge)^2 + \alpha_3 (\boldsymbol{\xi}^\wedge)^3 + \alpha_4 (\boldsymbol{\xi}^\wedge)^4, \quad (7.91)$$

where α_1 , α_2 , α_3 , and α_4 are unknown coefficients. We know that the

¹² This is a very lengthy derivation, but the result is exact.

series can be expressed using only terms up to quartic through the use of the identity in (7.62). Inserting this into (7.90), we have that

$$\mathcal{T} = \mathbf{1} + \boldsymbol{\xi}^\lambda + \alpha_1 (\boldsymbol{\xi}^\lambda)^2 + \alpha_2 (\boldsymbol{\xi}^\lambda)^3 + \alpha_3 (\boldsymbol{\xi}^\lambda)^4 + \alpha_4 (\boldsymbol{\xi}^\lambda)^5. \quad (7.92)$$

Using (7.62) to rewrite the quintic term using the lower-order terms, we have

$$\mathcal{T} = \mathbf{1} + (1 - \phi^4 \alpha_4) \boldsymbol{\xi}^\lambda + \alpha_1 (\boldsymbol{\xi}^\lambda)^2 + (\alpha_2 - 2\phi^2 \alpha_4) (\boldsymbol{\xi}^\lambda)^3 + \alpha_3 (\boldsymbol{\xi}^\lambda)^4. \quad (7.93)$$

Comparing the coefficients to those in (7.63), we can solve for α_1 , α_2 , α_3 , and α_4 such that

$$\begin{aligned} \mathcal{J}_\ell &= \mathbf{1} + \left(\frac{4 - \phi \sin \phi - 4 \cos \phi}{2\phi^2} \right) \boldsymbol{\xi}^\lambda + \left(\frac{4\phi - 5 \sin \phi + \phi \cos \phi}{2\phi^3} \right) (\boldsymbol{\xi}^\lambda)^2 \\ &\quad + \left(\frac{2 - \phi \sin \phi - 2 \cos \phi}{2\phi^4} \right) (\boldsymbol{\xi}^\lambda)^3 + \left(\frac{2\phi - 3 \sin \phi + \phi \cos \phi}{2\phi^5} \right) (\boldsymbol{\xi}^\lambda)^4. \end{aligned} \quad (7.94)$$

This avoids the need to work out \mathbf{J}_ℓ and \mathbf{Q}_ℓ individually and then assemble them into \mathcal{J}_ℓ .

Alternate expressions for the inverses are

$$\mathcal{J}_r^{-1} = \begin{bmatrix} \mathbf{J}_r^{-1} & -\mathbf{J}_r^{-1} \mathbf{Q}_r \mathbf{J}_r^{-1} \\ \mathbf{0} & \mathbf{J}_r^{-1} \end{bmatrix}, \quad (7.95a)$$

$$\mathcal{J}_\ell^{-1} = \begin{bmatrix} \mathbf{J}_\ell^{-1} & -\mathbf{J}_\ell^{-1} \mathbf{Q}_\ell \mathbf{J}_\ell^{-1} \\ \mathbf{0} & \mathbf{J}_\ell^{-1} \end{bmatrix}. \quad (7.95b)$$

We see that the singularities of \mathcal{J}_r and \mathcal{J}_ℓ are precisely the same as the singularities of \mathbf{J}_r and \mathbf{J}_ℓ , respectively, since

$$\det(\mathcal{J}_r) = (\det(\mathbf{J}_r))^2, \quad \det(\mathcal{J}_\ell) = (\det(\mathbf{J}_\ell))^2, \quad (7.96)$$

and having a non-zero determinant is a necessary and sufficient condition for invertibility (and therefore no singularity).

We also have that

$$\mathbf{T} = \begin{bmatrix} \mathbf{C} & \mathbf{J}_\ell \boldsymbol{\rho} \\ \mathbf{0}^T & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{C} & \mathbf{C} \mathbf{J}_r \boldsymbol{\rho} \\ \mathbf{0}^T & 1 \end{bmatrix}, \quad (7.97a)$$

$$\mathcal{T} = \begin{bmatrix} \mathbf{C} & (\mathbf{J}_\ell \boldsymbol{\rho})^\wedge \mathbf{C} \\ \mathbf{0} & \mathbf{C} \end{bmatrix} = \begin{bmatrix} \mathbf{C} & \mathbf{C} (\mathbf{J}_r \boldsymbol{\rho})^\wedge \mathbf{C} \\ \mathbf{0} & \mathbf{C} \end{bmatrix}, \quad (7.97b)$$

which tells us how to relate the $\boldsymbol{\rho}$ variable to the translational component of \mathbf{T} or \mathcal{T} .

It is also worth noting that $\mathcal{J} \mathcal{J}^T > 0$ (positive-definite) for either the left or right Jacobian. We can see this through the following factorization:

$$\mathcal{J} \mathcal{J}^T = \underbrace{\begin{bmatrix} \mathbf{1} & \mathbf{Q} \mathbf{J}^{-1} \\ \mathbf{0} & \mathbf{1} \end{bmatrix}}_{>0} \underbrace{\begin{bmatrix} \mathbf{J} \mathbf{J}^T & \mathbf{0} \\ \mathbf{0} & \mathbf{J} \mathbf{J}^T \end{bmatrix}}_{>0} \underbrace{\begin{bmatrix} \mathbf{1} & \mathbf{0} \\ \mathbf{J}^{-T} \mathbf{Q}^T & \mathbf{1} \end{bmatrix}}_{>0} > 0, \quad (7.98)$$

where we have used that $\mathbf{J}\mathbf{J}^T > 0$, which was shown previously.

Choosing the Left

In later sections and chapters, we will (arbitrarily) work with the left Jacobian, and it will therefore be useful to write out the BCH approximations in (7.75) and (7.83) using only the left Jacobian. For $SO(3)$, we have

$$\begin{aligned}\ln(\mathbf{C}_1\mathbf{C}_2)^\vee &= \ln(\exp(\phi_1^\wedge)\exp(\phi_2^\wedge))^\vee \\ &\approx \begin{cases} \mathbf{J}(\phi_2)^{-1}\phi_1 + \phi_2 & \text{if } \phi_1 \text{ small} \\ \phi_1 + \mathbf{J}(-\phi_1)^{-1}\phi_2 & \text{if } \phi_2 \text{ small} \end{cases}, \quad (7.99)\end{aligned}$$

where it is now implied that $\mathbf{J} = \mathbf{J}_\ell$, by convention¹³.

Similarly, for $SE(3)$, we have

$$\begin{aligned}\ln(\mathbf{T}_1\mathbf{T}_2)^\vee &= \ln(\exp(\xi_1^\wedge)\exp(\xi_2^\wedge))^\vee \\ &\approx \begin{cases} \mathcal{J}(\xi_2)^{-1}\xi_1 + \xi_2 & \text{if } \xi_1 \text{ small} \\ \xi_1 + \mathcal{J}(-\xi_1)^{-1}\xi_2 & \text{if } \xi_2 \text{ small} \end{cases}, \quad (7.100a)\end{aligned}$$

$$\begin{aligned}\ln(\mathcal{T}_1\mathcal{T}_2)^\vee &= \ln(\exp(\xi_1^\wedge)\exp(\xi_2^\wedge))^\vee \\ &\approx \begin{cases} \mathcal{J}(\xi_2)^{-1}\xi_1 + \xi_2 & \text{if } \xi_1 \text{ small} \\ \xi_1 + \mathcal{J}(-\xi_1)^{-1}\xi_2 & \text{if } \xi_2 \text{ small} \end{cases}, \quad (7.100b)\end{aligned}$$

where it is now implied that $\mathcal{J} = \mathcal{J}_\ell$, by convention.

7.1.6 Distance, Volume, Integration

We need to think about the concepts of distance, volume, and integration differently for Lie groups than for vectorspaces. This section quickly covers these topics for both rotations and poses.

Rotations

There are two common ways to define the difference of two rotations:

$$\phi_{12} = \ln(\mathbf{C}_1^T\mathbf{C}_2)^\vee, \quad (7.101a)$$

$$\phi_{21} = \ln(\mathbf{C}_2\mathbf{C}_1^T)^\vee, \quad (7.101b)$$

where $\mathbf{C}_1, \mathbf{C}_2 \in SO(3)$. One can be thought of as the right difference and the other the left. We can define the inner product for $\mathfrak{so}(3)$ as

$$\langle \phi_1^\wedge, \phi_2^\wedge \rangle = \frac{1}{2}\text{tr}(\phi_1^\wedge \phi_2^{\wedge T}) = \phi_1^T \phi_2. \quad (7.102)$$

The metric *distance* between two rotations can be thought of in two ways: (i) the square root of the inner product of the difference with

¹³ We will use this convention throughout the book and only show the subscript on the Jacobian when making specific points.

itself or (ii) the Euclidean norm of the difference:

$$\phi_{12} = \sqrt{\langle \ln(\mathbf{C}_1^T \mathbf{C}_2), \ln(\mathbf{C}_1^T \mathbf{C}_2) \rangle} = \sqrt{\langle \boldsymbol{\phi}_{12}^\wedge, \boldsymbol{\phi}_{12}^\wedge \rangle} = \sqrt{\boldsymbol{\phi}_{12}^T \boldsymbol{\phi}_{12}} = |\boldsymbol{\phi}_{12}|, \quad (7.103a)$$

$$\phi_{21} = \sqrt{\langle \ln(\mathbf{C}_2 \mathbf{C}_1^T), \ln(\mathbf{C}_2 \mathbf{C}_1^T) \rangle} = \sqrt{\langle \boldsymbol{\phi}_{21}^\wedge, \boldsymbol{\phi}_{21}^\wedge \rangle} = \sqrt{\boldsymbol{\phi}_{21}^T \boldsymbol{\phi}_{21}} = |\boldsymbol{\phi}_{21}|. \quad (7.103b)$$

This can also be viewed as the magnitude of the angle of the rotation difference.

To consider integrating functions of rotations, we parametrize $\mathbf{C} = \exp(\boldsymbol{\phi}^\wedge) \in SO(3)$. Perturbing $\boldsymbol{\phi}$ by a little bit results in the new rotation matrix, $\mathbf{C}' = \exp((\boldsymbol{\phi} + \delta\boldsymbol{\phi})^\wedge) \in SO(3)$. We have that the right and left differences (relative to \mathbf{C}) are

$$\begin{aligned} \ln(\delta\mathbf{C}_r)^\vee &= \ln(\mathbf{C}^T \mathbf{C}')^\vee = \ln(\mathbf{C}^T \exp((\boldsymbol{\phi} + \delta\boldsymbol{\phi})^\wedge))^\vee \\ &\approx \ln(\mathbf{C}^T \mathbf{C} \exp((\mathbf{J}_r \delta\boldsymbol{\phi})^\wedge))^\vee = \mathbf{J}_r \delta\boldsymbol{\phi}, \end{aligned} \quad (7.104a)$$

$$\begin{aligned} \ln(\delta\mathbf{C}_\ell)^\vee &= \ln(\mathbf{C}' \mathbf{C}^T)^\vee = \ln(\exp((\boldsymbol{\phi} + \delta\boldsymbol{\phi})^\wedge) \mathbf{C}^T)^\vee \\ &\approx \ln(\exp((\mathbf{J}_\ell \delta\boldsymbol{\phi})^\wedge) \mathbf{C} \mathbf{C}^T)^\vee = \mathbf{J}_\ell \delta\boldsymbol{\phi}, \end{aligned} \quad (7.104b)$$

where \mathbf{J}_r and \mathbf{J}_ℓ are evaluated at $\boldsymbol{\phi}$. To compute the infinitesimal volume element, we want to find the volume of the parallelepiped formed by the columns of \mathbf{J}_r or \mathbf{J}_ℓ , which is simply the corresponding determinant¹⁴:

$$d\mathbf{C}_r = |\det(\mathbf{J}_r)| d\boldsymbol{\phi}, \quad (7.105a)$$

$$d\mathbf{C}_\ell = |\det(\mathbf{J}_\ell)| d\boldsymbol{\phi}. \quad (7.105b)$$

We note that

$$\det(\mathbf{J}_\ell) = \det(\mathbf{C} \mathbf{J}_r) = \underbrace{\det(\mathbf{C})}_1 \det(\mathbf{J}_r) = \det(\mathbf{J}_r), \quad (7.106)$$

which means that regardless of which distance metric we use, right or left, the infinitesimal volume element is the same. This is true for all *unimodular* Lie groups, such as $SO(3)$. Therefore, we can write

$$d\mathbf{C} = |\det(\mathbf{J})| d\boldsymbol{\phi}, \quad (7.107)$$

for the calculation of an infinitesimal volume element.

It turns out that

$$\begin{aligned} |\det(\mathbf{J})| &= 2 \frac{1 - \cos \boldsymbol{\phi}}{\boldsymbol{\phi}^2} = \frac{2}{\boldsymbol{\phi}^2} \left(1 - 1 + \frac{\boldsymbol{\phi}^2}{2!} - \frac{\boldsymbol{\phi}^4}{4!} + \frac{\boldsymbol{\phi}^6}{6!} - \frac{\boldsymbol{\phi}^8}{8!} + \dots \right) \\ &= 1 - \frac{1}{12} \boldsymbol{\phi}^2 + \frac{1}{360} \boldsymbol{\phi}^4 - \frac{1}{20160} \boldsymbol{\phi}^6 + \dots, \end{aligned} \quad (7.108)$$

¹⁴ We are slightly abusing notation here by writing $d\mathbf{C}$, but hopefully it is clear from context what is meant.

where $\phi = |\boldsymbol{\phi}|$. For most practical situations we can safely use just the first two or even one term of this expression.

Integrating functions of rotations can then be carried out like this:

$$\int_{SO(3)} f(\mathbf{C}) d\mathbf{C} \rightarrow \int_{|\boldsymbol{\phi}| < \pi} f(\boldsymbol{\phi}) |\det(\mathbf{J})| d\boldsymbol{\phi}, \quad (7.109)$$

where we are careful to ensure $|\boldsymbol{\phi}| < \pi$ so as to sweep out all of $SO(3)$ just once (due to the surjective-only nature of the exponential map).

Poses

We briefly summarize the $SE(3)$ and $\text{Ad}(SE(3))$ results as they are very similar to $SO(3)$. We can define right and left distance metrics:

$$\xi_{12} = \ln (\mathbf{T}_1^{-1} \mathbf{T}_2)^\vee = \ln (\mathcal{T}_1^{-1} \mathcal{T}_2)^\vee, \quad (7.110a)$$

$$\xi_{21} = \ln (\mathbf{T}_2 \mathbf{T}_1^{-1})^\vee = \ln (\mathcal{T}_2 \mathcal{T}_1^{-1})^\vee. \quad (7.110b)$$

The 4×4 and 6×6 inner products are

$$\langle \boldsymbol{\xi}_1^\wedge, \boldsymbol{\xi}_2^\wedge \rangle = -\text{tr} \left(\boldsymbol{\xi}_1^\wedge \begin{bmatrix} \frac{1}{2}\mathbf{1} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} \boldsymbol{\xi}_2^{\wedge T} \right) = \boldsymbol{\xi}_1^T \boldsymbol{\xi}_2, \quad (7.111a)$$

$$\langle \boldsymbol{\xi}_1^\wedge, \boldsymbol{\xi}_2^\wedge \rangle = -\text{tr} \left(\boldsymbol{\xi}_1^\wedge \begin{bmatrix} \frac{1}{4}\mathbf{1} & \mathbf{0} \\ \mathbf{0} & \frac{1}{2}\mathbf{1} \end{bmatrix} \boldsymbol{\xi}_2^{\wedge T} \right) = \boldsymbol{\xi}_1^T \boldsymbol{\xi}_2. \quad (7.111b)$$

Note that we could adjust the weighting matrix in the middle to weight rotation and translation differently if we so desired. The right and left distances are

$$\xi_{12} = \sqrt{\langle \boldsymbol{\xi}_{12}^\wedge, \boldsymbol{\xi}_{12}^\wedge \rangle} = \sqrt{\langle \boldsymbol{\xi}_{12}^\wedge, \boldsymbol{\xi}_{12}^\wedge \rangle} = \sqrt{\boldsymbol{\xi}_{12}^T \boldsymbol{\xi}_{12}} = |\boldsymbol{\xi}_{12}|, \quad (7.112a)$$

$$\xi_{21} = \sqrt{\langle \boldsymbol{\xi}_{21}^\wedge, \boldsymbol{\xi}_{21}^\wedge \rangle} = \sqrt{\langle \boldsymbol{\xi}_{21}^\wedge, \boldsymbol{\xi}_{21}^\wedge \rangle} = \sqrt{\boldsymbol{\xi}_{21}^T \boldsymbol{\xi}_{21}} = |\boldsymbol{\xi}_{21}|. \quad (7.112b)$$

Using the parametrization

$$\mathbf{T} = \exp(\boldsymbol{\xi}^\wedge) \quad (7.113)$$

and the perturbation

$$\mathbf{T}' = \exp((\boldsymbol{\xi} + \delta\boldsymbol{\xi})^\wedge), \quad (7.114)$$

the differences (relative to \mathbf{T}) are

$$\ln(\delta\mathbf{T}_r)^\vee = \ln(\mathbf{T}^{-1}\mathbf{T}')^\vee \approx \mathcal{J}_r \delta\boldsymbol{\xi}, \quad (7.115a)$$

$$\ln(\delta\mathbf{T}_\ell)^\vee = \ln(\mathbf{T}'\mathbf{T}^{-1})^\vee \approx \mathcal{J}_\ell \delta\boldsymbol{\xi}. \quad (7.115b)$$

The right and left infinitesimal volume elements are

$$d\mathbf{T}_r = |\det(\mathcal{J}_r)| d\boldsymbol{\xi}, \quad (7.116a)$$

$$d\mathbf{T}_\ell = |\det(\mathcal{J}_\ell)| d\boldsymbol{\xi}. \quad (7.116b)$$

We have that

$$\det(\mathcal{J}_\ell) = \det(\mathcal{T}\mathcal{J}_r) = \det(\mathcal{T}) \det(\mathcal{J}_r) = \det(\mathcal{J}_r), \quad (7.117)$$

since $\det(\mathcal{T}) = (\det(\mathbf{C}))^2 = 1$. We can therefore write

$$d\mathbf{T} = |\det(\mathcal{J})| d\boldsymbol{\xi} \quad (7.118)$$

for our integration volume. Finally, we have that

$$\begin{aligned} |\det(\mathcal{J})| &= |\det(\mathbf{J})|^2 = \left(2 \frac{1 - \cos \phi}{\phi^2}\right)^2 \\ &= 1 - \frac{1}{6}\phi^2 + \frac{1}{80}\phi^4 - \frac{17}{30240}\phi^6 + \dots, \end{aligned} \quad (7.119)$$

and again we probably will never need more than two terms of this expression.

To integrate functions over $SE(3)$, we can now use our infinitesimal volume in the calculation:

$$\int_{SE(3)} f(\mathbf{T}) d\mathbf{T} = \int_{\mathbb{R}^3, |\phi| < \pi} f(\boldsymbol{\xi}) |\det(\mathcal{J})| d\boldsymbol{\xi}, \quad (7.120)$$

where we limit ϕ to the ball of radius π (due to the surjective-only nature of the exponential map) but let $\boldsymbol{\rho} \in \mathbb{R}^3$.

7.1.7 Interpolation

We will have occasion later to interpolate between two elements of a matrix Lie group. Unfortunately, the typical linear interpolation scheme,

$$x = (1 - \alpha)x_1 + \alpha x_2, \quad \alpha \in [0, 1], \quad (7.121)$$

will not work because this interpolation scheme does not satisfy closure (i.e., the result is no longer in the group). In other words,

$$(1 - \alpha)\mathbf{C}_1 + \alpha\mathbf{C}_2 \notin SO(3), \quad (7.122a)$$

$$(1 - \alpha)\mathbf{T}_1 + \alpha\mathbf{T}_2 \notin SE(3) \quad (7.122b)$$

for some values of $\alpha \in [0, 1]$ with $\mathbf{C}_1, \mathbf{C}_2 \in SO(3)$, $\mathbf{T}_1, \mathbf{T}_2 \in SE(3)$. We must rethink what interpolation means for Lie groups.

Rotations

There are many possible interpolation schemes that we could define. One of these is the following:

$$\mathbf{C} = (\mathbf{C}_2 \mathbf{C}_1^T)^\alpha \mathbf{C}_1, \quad \alpha \in [0, 1], \quad (7.123)$$

where $\mathbf{C}, \mathbf{C}_1, \mathbf{C}_2 \in SO(3)$. We see that when $\alpha = 0$, we have $\mathbf{C} = \mathbf{C}_1$, and when $\alpha = 1$, we have \mathbf{C}_2 . The nice thing about this scheme is that

we guarantee closure, meaning $\mathbf{C} \in SO(3)$ for all $\alpha \in [0, 1]$. This is because we know that $\mathbf{C}_{21} = \exp(\phi^\wedge) = \mathbf{C}_2 \mathbf{C}_1^T$ is still a rotation matrix due to closure of the Lie group. Exponentiating by the interpolation variable keeps the result in $SO(3)$,

$$\mathbf{C}_{21}^\alpha = \exp(\phi^\wedge)^\alpha = \exp(\alpha \phi^\wedge) \in SO(3), \quad (7.124)$$

and finally, compounding with \mathbf{C}_1 results in a member of $SO(3)$, again due to closure of the group. We can also see that we are essentially just scaling the rotation angle of \mathbf{C}_{21} by α , which is appealing intuitively.

Our scheme in (7.123) is actually similar to (7.121), if we rearrange it a bit:

$$x = \alpha(x_2 - x_1) + x_1. \quad (7.125)$$

Or, letting $x = \ln(y)$, $x_1 = \ln(y_1)$, $x_2 = \ln(y_2)$, we can rewrite it as

$$y = (y_2 y_1^{-1})^\alpha y_1, \quad (7.126)$$

which is very similar to our proposed scheme. Given our understanding of the relationship between $\mathfrak{so}(3)$ and $SO(3)$ (i.e., through the exponential map), it is therefore not a leap to understand that (7.123) is somehow defining linear-like interpolation in the Lie algebra, where we can treat elements as vectors.

To examine this further, we let $\mathbf{C} = \exp(\varphi^\wedge)$, $\mathbf{C}_1 = \exp(\phi_1^\wedge)$, $\mathbf{C}_2 = \exp(\phi_2^\wedge) \in SO(3)$ with $\varphi, \phi_1, \phi_2 \in \mathbb{R}^3$. If we are able to make the assumption that ϕ is small (in the sense of distance from the previous section), then we have

$$\begin{aligned} \varphi &= \ln(\mathbf{C})^\vee = \ln\left((\mathbf{C}_2 \mathbf{C}_1^T)^\alpha \mathbf{C}_1\right)^\vee \\ &= \ln(\exp(\alpha \phi^\wedge) \exp(\phi_1^\wedge))^\vee \approx \alpha \mathbf{J}(\phi_1)^{-1} \phi + \phi_1, \end{aligned} \quad (7.127)$$

which is comparable to (7.125) and is a form of linear interpolation. Another case worth noting is when $\mathbf{C}_1 = \mathbf{1}$, whereupon

$$\mathbf{C} = \mathbf{C}_2^\alpha, \quad \varphi = \alpha \phi_2, \quad (7.128)$$

with no approximation.

Another way to interpret our interpolation scheme is that it is enforcing a constant angular velocity, $\boldsymbol{\omega}$. If we think of our rotation matrix as being a function of time, $\mathbf{C}(t)$, then the scheme is

$$\mathbf{C}(t) = (\mathbf{C}(t_2) \mathbf{C}(t_1)^T)^\alpha \mathbf{C}(t_1), \quad \alpha = \frac{t - t_1}{t_2 - t_1}. \quad (7.129)$$

Defining the constant angular velocity as

$$\boldsymbol{\omega} = \frac{1}{t_2 - t_1} \phi, \quad (7.130)$$

the scheme becomes

$$\mathbf{C}(t) = \exp((t - t_1)\boldsymbol{\omega}^\wedge) \mathbf{C}(t_1). \quad (7.131)$$

This is exactly the solution to Poisson's equation, (6.45),

$$\dot{\mathbf{C}}(t) = \boldsymbol{\omega}^\wedge \mathbf{C}(t), \quad (7.132)$$

with constant angular velocity¹⁵. Thus, while other interpolation schemes are possible, this one has a strong physical connection.

Perturbed Rotations

Another thing that will be very useful to investigate, is what happens to \mathbf{C} if we perturb \mathbf{C}_1 and/or \mathbf{C}_2 a little bit. Suppose now that $\mathbf{C}', \mathbf{C}'_1, \mathbf{C}'_2 \in SO(3)$ are the perturbed rotation matrices with the (left) differences¹⁶ given as

$$\delta\boldsymbol{\varphi} = \ln(\mathbf{C}'\mathbf{C}^T)^\vee, \quad \delta\boldsymbol{\phi}_1 = \ln(\mathbf{C}'_1\mathbf{C}_1^T)^\vee, \quad \delta\boldsymbol{\phi}_2 = \ln(\mathbf{C}'_2\mathbf{C}_2^T)^\vee. \quad (7.133)$$

The interpolation scheme must hold for the perturbed rotation matrices:

$$\mathbf{C}' = \left(\mathbf{C}'_2\mathbf{C}'_1^T\right)^\alpha \mathbf{C}'_1, \quad \alpha \in [0, 1]. \quad (7.134)$$

We are interested in finding a relationship between $\delta\boldsymbol{\varphi}$ and $\delta\boldsymbol{\phi}_1, \delta\boldsymbol{\phi}_2$. Substituting in our perturbations we have

$$\begin{aligned} \exp(\delta\boldsymbol{\varphi}^\wedge) \mathbf{C} &= \underbrace{\left(\exp(\delta\boldsymbol{\phi}_2^\wedge) \mathbf{C}_2 \mathbf{C}_1^T \exp(-\delta\boldsymbol{\phi}_1^\wedge)\right)}_{\approx \exp((\boldsymbol{\phi} + \mathbf{J}(\boldsymbol{\phi})^{-1}(\delta\boldsymbol{\phi}_2 - \mathbf{C}_{21}\delta\boldsymbol{\phi}_1))^\wedge)}^\alpha \exp(\delta\boldsymbol{\phi}_1^\wedge) \mathbf{C}_1, \end{aligned} \quad (7.135)$$

where we have assumed the perturbations are small to make the approximation hold inside the brackets. Bringing the interpolation variable inside the exponential, we have

$$\begin{aligned} &\exp(\delta\boldsymbol{\varphi}^\wedge) \mathbf{C} \\ &\approx \underbrace{\exp\left((\alpha\boldsymbol{\phi} + \alpha\mathbf{J}(\boldsymbol{\phi})^{-1}(\delta\boldsymbol{\phi}_2 - \mathbf{C}_{21}\delta\boldsymbol{\phi}_1))^\wedge\right)}_{\approx \exp((\alpha\mathbf{J}(\alpha\boldsymbol{\phi})\mathbf{J}(\boldsymbol{\phi})^{-1}(\delta\boldsymbol{\phi}_2 - \mathbf{C}_{21}\delta\boldsymbol{\phi}_1))^\wedge)} \exp(\delta\boldsymbol{\phi}_1^\wedge) \mathbf{C}_1 \\ &\approx \exp\left((\alpha\mathbf{J}(\alpha\boldsymbol{\phi})\mathbf{J}(\boldsymbol{\phi})^{-1}(\delta\boldsymbol{\phi}_2 - \mathbf{C}_{21}\delta\boldsymbol{\phi}_1))^\wedge\right) \\ &\quad \times \underbrace{\exp((\mathbf{C}_{21}^\alpha\delta\boldsymbol{\phi}_1)^\wedge)}_{\mathbf{C}} \underbrace{\mathbf{C}_{21}^\alpha \mathbf{C}_1}. \end{aligned} \quad (7.136)$$

¹⁵ Kinematics will be discussed in further detail later in this chapter.

¹⁶ In anticipation of how we will use this result, we will consider perturbations on the left, but we saw in the previous section that there are equivalent perturbations on the right and in the middle.

Dropping the \mathbf{C} from both sides, expanding the matrix exponentials, distributing the multiplication, and then keeping only first-order terms in the perturbation quantities, we have

$$\delta\varphi = \alpha \mathbf{J}(\alpha\phi)\mathbf{J}(\phi)^{-1}(\delta\phi_2 - \mathbf{C}_{21}\delta\phi_1) + \mathbf{C}_{21}^\alpha\delta\phi_1. \quad (7.137)$$

Manipulating a little further (using several identities involving the Jacobians), we can show that this simplifies to

$$\delta\varphi = (\mathbf{1} - \mathbf{A}(\alpha, \phi))\delta\phi_1 + \mathbf{A}(\alpha, \phi)\delta\phi_2, \quad (7.138)$$

where

$$\mathbf{A}(\alpha, \phi) = \alpha \mathbf{J}(\alpha\phi)\mathbf{J}(\phi)^{-1}. \quad (7.139)$$

We see that this has a very nice form that mirrors the usual linear interpolation scheme. Notably, when ϕ is small, then $\mathbf{A}(\alpha, \phi) \approx \alpha \mathbf{1}$.

Although we have a means of computing $\mathbf{A}(\alpha, \phi)$ in closed form (via $\mathbf{J}(\cdot)$), we can work out a series expression for it as well. In terms of our series expressions for $\mathbf{J}(\cdot)$ and its inverse, we have

$$\mathbf{A}(\alpha, \phi) = \underbrace{\alpha \left(\sum_{k=0}^{\infty} \frac{1}{(k+1)!} \alpha^k (\phi^\wedge)^k \right)}_{\mathbf{J}(\alpha\phi)} \underbrace{\left(\sum_{\ell=0}^{\infty} \frac{B_\ell}{\ell!} (\phi^\wedge)^\ell \right)}_{\mathbf{J}(\phi)^{-1}}. \quad (7.140)$$

We can use a discrete convolution, or *Cauchy product* (of two series), to rewrite this as

$$\mathbf{A}(\alpha, \phi) = \sum_{n=0}^{\infty} \frac{F_n(\alpha)}{n!} (\phi^\wedge)^n, \quad (7.141)$$

where

$$F_n(\alpha) = \frac{1}{n+1} \sum_{m=0}^n \binom{n+1}{m} B_m \alpha^{n+1-m} = \sum_{\beta=0}^{\alpha-1} \beta^n, \quad (7.142)$$

is a version of *Faulhaber's formula*. The first few Faulhaber coefficients (as we will call them) are

$$\begin{aligned} F_0(\alpha) &= \alpha, \quad F_1(\alpha) = \frac{\alpha(\alpha-1)}{2}, \quad F_2(\alpha) = \frac{\alpha(\alpha-1)(2\alpha-1)}{6}, \\ F_3(\alpha) &= \frac{\alpha^2(\alpha-1)^2}{4}, \quad \dots \end{aligned} \quad (7.143)$$

Putting these back into $\mathbf{A}(\alpha, \phi)$, we have

$$\begin{aligned} \mathbf{A}(\alpha, \phi) &= \alpha \mathbf{1} + \frac{\alpha(\alpha-1)}{2} \phi^\wedge + \frac{\alpha(\alpha-1)(2\alpha-1)}{12} \phi^\wedge \phi^\wedge \\ &\quad + \frac{\alpha^2(\alpha-1)^2}{24} \phi^\wedge \phi^\wedge \phi^\wedge + \dots, \end{aligned} \quad (7.144)$$

where we likely would not need many terms if ϕ is small.

Baron Augustin-Louis Cauchy (1789-1857) was a French mathematician who pioneered the study of continuity in terms of infinitesimals, almost singlehandedly founded complex analysis, and initiated the study of permutation groups in abstract algebra.

Johann Faulhaber (1580-1635) was a German mathematician whose major contribution involved calculating the sums of powers of integers. Jakob Bernoulli makes references to Faulhaber in his *Ars Conjectandi*.

Alternate Interpretation of Perturbed Rotations

Technically speaking, the last sum on the far right of (7.142) does not make much sense since $\alpha \in [0, 1]$, but we can also get to this another way. Let us pretend for the moment that α is in fact a positive integer. Then we can expand the exponentiated part of our interpolation formula according to

$$(\exp(\delta\phi^\wedge) \mathbf{C})^\alpha = \underbrace{\exp(\delta\phi^\wedge) \mathbf{C} \cdots \exp(\delta\phi^\wedge) \mathbf{C}}_\alpha, \quad (7.145)$$

where $\mathbf{C} = \exp(\phi^\wedge)$. We can then move all of the $\delta\phi$ terms to the far left so that

$$(\exp(\delta\phi^\wedge) \mathbf{C})^\alpha = \exp(\delta\phi^\wedge) \exp((\mathbf{C} \delta\phi)^\wedge) \cdots \exp((\mathbf{C}^{\alpha-1} \delta\phi)^\wedge) \mathbf{C}^\alpha, \quad (7.146)$$

where we have not yet made any approximations. Expanding each of the exponentials, multiplying out, and keeping only terms first-order in $\delta\phi$ leaves us with

$$\begin{aligned} (\exp(\delta\phi^\wedge) \mathbf{C})^\alpha &\approx \left(\mathbf{1} + \left(\left(\sum_{\beta=0}^{\alpha-1} \mathbf{C}^\beta \right) \delta\phi \right)^\wedge \right) \mathbf{C}^\alpha \\ &= (\mathbf{1} + (\mathbf{A}(\alpha, \phi) \delta\phi)^\wedge) \mathbf{C}^\alpha, \end{aligned} \quad (7.147)$$

where

$$\begin{aligned} \mathbf{A}(\alpha, \phi) &= \sum_{\beta=0}^{\alpha-1} \mathbf{C}^\beta = \sum_{\beta=0}^{\alpha-1} \exp(\beta\phi^\wedge) = \sum_{\beta=0}^{\alpha-1} \sum_{n=0}^{\infty} \frac{1}{n!} \beta^n (\phi^\wedge)^n \\ &= \sum_{n=0}^{\infty} \frac{1}{n!} \underbrace{\left(\sum_{\beta=0}^{\alpha-1} \beta^n \right)}_{F_n(\alpha)} (\phi^\wedge)^n = \sum_{n=0}^{\infty} \frac{F_n(\alpha)}{n!} (\phi^\wedge)^n, \end{aligned} \quad (7.148)$$

which is the same as (7.141). Some examples of Faulhaber's coefficients are:

$$F_0(\alpha) = 0^0 + 1^0 + 2^0 + \cdots + (\alpha - 1)^0 = \alpha, \quad (7.149a)$$

$$F_1(\alpha) = 0^1 + 1^1 + 2^1 + \cdots + (\alpha - 1)^1 = \frac{\alpha(\alpha - 1)}{2}, \quad (7.149b)$$

$$F_2(\alpha) = 0^2 + 1^2 + 2^2 + \cdots + (\alpha - 1)^2 = \frac{\alpha(\alpha - 1)(2\alpha - 1)}{6}, \quad (7.149c)$$

$$F_3(\alpha) = 0^3 + 1^3 + 2^3 + \cdots + (\alpha - 1)^3 = \frac{\alpha^2(\alpha - 1)^2}{4}, \quad (7.149d)$$

which are the same as what we had before. Interestingly, these expressions work even when $\alpha \in [0, 1]$.

Poses

Interpolation for elements of $SE(3)$ parallels the $SO(3)$ case. We define the interpolation scheme as the following:

$$\mathbf{T} = (\mathbf{T}_2 \mathbf{T}_1^{-1})^\alpha \mathbf{T}_1, \quad \alpha \in [0, 1]. \quad (7.150)$$

Again, this scheme ensures that $\mathbf{T} = \exp(\zeta^\wedge) \in SE(3)$ as long as $\mathbf{T}_1 = \exp(\xi_1^\wedge), \mathbf{T}_2 = \exp(\xi_2^\wedge) \in SE(3)$. Let $\mathbf{T}_{21} = \mathbf{T}_2 \mathbf{T}_1^{-1} = \exp(\xi^\wedge)$, so that

$$\begin{aligned} \zeta &= \ln(\mathbf{T})^\vee = \ln((\mathbf{T}_2 \mathbf{T}_1^{-1})^\alpha \mathbf{T}_1)^\vee = \ln(\exp(\alpha \xi^\wedge) \exp(\xi_1^\wedge))^\vee \\ &\approx \alpha \mathcal{J}(\xi_1) \mathcal{J}(\xi)^{-1} \xi + \xi_1, \end{aligned} \quad (7.151)$$

where the approximation on the right holds if ξ is small. When $\mathbf{T}_1 = \mathbf{1}$, the scheme becomes

$$\mathbf{T} = \mathbf{T}_2^\alpha, \quad \zeta = \alpha \xi_2, \quad (7.152)$$

with no approximation.

Perturbed Poses

As in the $SO(3)$ case, it will be useful to investigate what happens to \mathbf{T} if we perturb \mathbf{T}_1 and/or \mathbf{T}_2 a little bit. Suppose now that $\mathbf{T}', \mathbf{T}'_1, \mathbf{T}'_2 \in SE(3)$ are the perturbed transformation matrices with the (left) differences given as

$$\delta\zeta = \ln(\mathbf{T}' \mathbf{T}^{-1})^\vee, \quad \delta\xi_1 = \ln(\mathbf{T}'_1 \mathbf{T}_1^{-1})^\vee, \quad \delta\xi_2 = \ln(\mathbf{T}'_2 \mathbf{T}_2^{-1})^\vee. \quad (7.153)$$

The interpolation scheme must hold for the perturbed transformation matrices:

$$\mathbf{T}' = (\mathbf{T}'_2 \mathbf{T}'_1^{-1})^\alpha \mathbf{T}'_1, \quad \alpha \in [0, 1]. \quad (7.154)$$

We are interested in finding a relationship between $\delta\zeta$ and $\delta\xi_1, \delta\xi_2$.

The derivation is very similar to $SO(3)$, so we will simply state the result:

$$\delta\zeta = (\mathbf{1} - \mathcal{A}(\alpha, \xi)) \delta\xi_1 + \mathcal{A}(\alpha, \xi) \delta\xi_2, \quad (7.155)$$

where

$$\mathcal{A}(\alpha, \xi) = \alpha \mathcal{J}(\alpha \xi) \mathcal{J}(\xi)^{-1}, \quad (7.156)$$

and we note this is a 6×6 matrix. Again, we see this has a very nice form that mirrors the usual linear interpolation scheme. Notably, when ξ is small, then $\mathcal{A}(\alpha, \xi) \approx \alpha \mathbf{1}$. In series form, we have

$$\mathcal{A}(\alpha, \xi) = \sum_{n=0}^{\infty} \frac{F_n(\alpha)}{n!} (\xi^\wedge)^n, \quad (7.157)$$

where the $F_n(\alpha)$ are the Faulhaber coefficients discussed earlier.

7.1.8 Homogeneous Points

As discussed in Section 6.3.1, points in \mathbb{R}^3 can be represented using 4×1 *homogeneous coordinates* (Hartley and Zisserman, 2000), as follows:

$$\mathbf{p} = \begin{bmatrix} sx \\ sy \\ sz \\ s \end{bmatrix} = \begin{bmatrix} \boldsymbol{\varepsilon} \\ \eta \end{bmatrix},$$

where s is some real, nonzero scalar, $\boldsymbol{\varepsilon} \in \mathbb{R}^3$, and η is scalar. When s is zero, it is not possible to convert back to \mathbb{R}^3 , as this case represents points that are infinitely far away. Thus, homogeneous coordinates can be used to describe near and distant landmarks with no singularities or scaling issues (Triggs et al., 2000). They are also a natural representation in that points may then be transformed from one frame to another very easily using transformation matrices (e.g., $\mathbf{p}_2 = \mathbf{T}_{21} \mathbf{p}_1$).

We will later make use of the following two operators¹⁷ for manipulating 4×1 columns:

$$\begin{bmatrix} \boldsymbol{\varepsilon} \\ \eta \end{bmatrix}^\odot = \begin{bmatrix} \eta \mathbf{1} & -\boldsymbol{\varepsilon}^\wedge \\ \mathbf{0}^T & \mathbf{0}^T \end{bmatrix}, \quad \begin{bmatrix} \boldsymbol{\varepsilon} \\ \eta \end{bmatrix}^\odot = \begin{bmatrix} \mathbf{0} & \boldsymbol{\varepsilon} \\ -\boldsymbol{\varepsilon}^\wedge & \mathbf{0} \end{bmatrix}, \quad (7.158)$$

which result in a 4×6 and 6×4 , respectively. With these definitions, we have the following useful identities:

$$\boldsymbol{\xi}^\wedge \mathbf{p} \equiv \mathbf{p}^\odot \boldsymbol{\xi}, \quad \mathbf{p}^T \boldsymbol{\xi}^\wedge \equiv \boldsymbol{\xi}^T \mathbf{p}^\odot, \quad (7.159)$$

where $\boldsymbol{\xi} \in \mathbb{R}^6$ and $\mathbf{p} \in \mathbb{R}^4$, which will prove useful when manipulating expressions involving points and poses together. We also have the identity,

$$(\mathbf{T}\mathbf{p})^\odot \equiv \mathbf{T}\mathbf{p}^\odot \mathcal{T}^{-1}, \quad (7.160)$$

which is similar to some others we have already seen.

7.1.9 Calculus and Optimization

Now that we have introduced homogeneous points, we formulate a bit of calculus to allow us to optimize functions of rotations and/or poses, sometimes in combination with three-dimensional points. As usual, we first study rotations and then poses. Absil et al. (2009) provides a much more detailed look at how to carry out optimization on matrix manifolds, exploring first-order, second-order, and trust-region methods.

¹⁷ The \odot operator for 4×1 columns is similar to the \boxdot operator defined by Furgale (2011), which did not have the negative sign.

Rotations

We have already seen in Section 6.2.5 a preview of perturbing expressions in terms of their Euler angles. We first consider directly taking the Jacobian of a rotated point with respect to the Lie algebra vector representing the rotation:

$$\frac{\partial(\mathbf{C}\mathbf{v})}{\partial\phi}, \quad (7.161)$$

where $\mathbf{C} = \exp(\phi^\wedge) \in SO(3)$ and $\mathbf{v} \in \mathbb{R}^3$ is some arbitrary three-dimensional point.

To do this, we can start by taking the derivative with respect to a single element of $\phi = (\phi_1, \phi_2, \phi_3)$. Applying the definition of a derivative along the $\mathbf{1}_i$ direction, we have

$$\frac{\partial(\mathbf{C}\mathbf{v})}{\partial\phi_i} = \lim_{h \rightarrow 0} \frac{\exp((\phi + h\mathbf{1}_i)^\wedge)\mathbf{v} - \exp(\phi^\wedge)\mathbf{v}}{h}, \quad (7.162)$$

which we will refer to as a *directional derivative*. Since we are interested in the limit of h infinitely small, we can use the approximate BCH formula to write

$$\begin{aligned} \exp((\phi + h\mathbf{1}_i)^\wedge) &\approx \exp((\mathbf{J}h\mathbf{1}_i)^\wedge)\exp(\phi^\wedge) \\ &\approx (\mathbf{1} + h(\mathbf{J}\mathbf{1}_i)^\wedge)\exp(\phi^\wedge), \end{aligned} \quad (7.163)$$

where \mathbf{J} is the (left) Jacobian of $SO(3)$, evaluated at ϕ . Plugging this back into (7.162), we find that

$$\frac{\partial(\mathbf{C}\mathbf{v})}{\partial\phi_i} = (\mathbf{J}\mathbf{1}_i)^\wedge\mathbf{C}\mathbf{v} = -(\mathbf{C}\mathbf{v})^\wedge\mathbf{J}\mathbf{1}_i. \quad (7.164)$$

Stacking the three directional derivatives alongside one another provides the desired Jacobian:

$$\frac{\partial(\mathbf{C}\mathbf{v})}{\partial\phi} = -(\mathbf{C}\mathbf{v})^\wedge\mathbf{J}. \quad (7.165)$$

Moreover, if $\mathbf{C}\mathbf{v}$ appears inside another scalar function, $u(\mathbf{x})$, with $\mathbf{x} = \mathbf{C}\mathbf{v}$, then we have

$$\frac{\partial u}{\partial\phi} = \frac{\partial u}{\partial\mathbf{x}} \frac{\partial\mathbf{x}}{\partial\phi} = -\frac{\partial u}{\partial\mathbf{x}} (\mathbf{C}\mathbf{v})^\wedge\mathbf{J}, \quad (7.166)$$

by the chain rule of differentiation. The result is the transpose of the gradient of u with respect to ϕ .

If we wanted to perform simple gradient descent of our function, we could take a step in the direction of the negative gradient, evaluated at our linearization point, $\mathbf{C}_{\text{op}} = \exp(\phi_{\text{op}}^\wedge)$:

$$\phi = \phi_{\text{op}} - \alpha \underbrace{\mathbf{J}^T (\mathbf{C}_{\text{op}}\mathbf{v})^\wedge \frac{\partial u}{\partial\mathbf{x}} \Big|_{\mathbf{x}=\mathbf{C}_{\text{op}}\mathbf{v}}}_\delta^T, \quad (7.167)$$

where $\alpha > 0$ defines the step size.

We can easily see that stepping in this direction (by a small amount) will reduce the function value:

$$u(\exp(\phi^\wedge)\mathbf{v}) - u(\exp(\phi_{\text{op}}^\wedge)\mathbf{v}) \approx -\underbrace{\alpha \boldsymbol{\delta}^T (\mathbf{J}\mathbf{J}^T) \boldsymbol{\delta}}_{\geq 0}. \quad (7.168)$$

However, this is not the most streamlined way we could optimize u with respect to \mathbf{C} because it requires that we store our rotation as a rotation vector, ϕ , which has singularities associated with it. Plus, we need to compute the Jacobian matrix, \mathbf{J} .

A cleaner way to carry out optimization is to find an update step for \mathbf{C} in the form of a small rotation on the left¹⁸ rather than directly on the Lie algebra rotation vector representing \mathbf{C} :

$$\mathbf{C} = \exp(\psi^\wedge) \mathbf{C}_{\text{op}}. \quad (7.169)$$

The previous update can actually be cast in this form by using the approximate BCH formula once again:

$$\begin{aligned} \mathbf{C} = \exp(\phi^\wedge) &= \exp\left((\phi_{\text{op}} - \alpha \mathbf{J}^T \boldsymbol{\delta})^\wedge\right) \\ &\approx \exp\left(-\alpha (\mathbf{J}\mathbf{J}^T \boldsymbol{\delta})^\wedge\right) \mathbf{C}_{\text{op}}, \end{aligned} \quad (7.170)$$

or in other words, we could let $\psi = -\alpha \mathbf{J}\mathbf{J}^T \boldsymbol{\delta}$ to accomplish the same thing as before, but this still requires that we compute \mathbf{J} . Instead, we can essentially just drop $\mathbf{J}\mathbf{J}^T > 0$ from the update and use

$$\psi = -\alpha \boldsymbol{\delta}, \quad (7.171)$$

which still reduces the function,

$$u(\mathbf{C}\mathbf{v}) - u(\mathbf{C}_{\text{op}}\mathbf{v}) \approx -\underbrace{\alpha \boldsymbol{\delta}^T \boldsymbol{\delta}}_{\geq 0}, \quad (7.172)$$

but takes a slightly different direction to do so.

Another way to look at this is that we are computing the Jacobian with respect to ψ , where the perturbation is applied on the left¹⁹. Along the ψ_i direction, we have

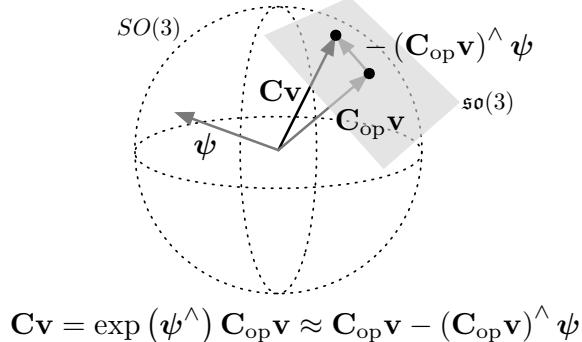
$$\begin{aligned} \frac{\partial(\mathbf{C}\mathbf{v})}{\partial\psi_i} &= \lim_{h \rightarrow 0} \frac{\exp(h\mathbf{1}_i^\wedge)\mathbf{C}\mathbf{v} - \mathbf{C}\mathbf{v}}{h} \\ &\approx \lim_{h \rightarrow 0} \frac{(\mathbf{1} + h\mathbf{1}_i^\wedge)\mathbf{C}\mathbf{v} - \mathbf{C}\mathbf{v}}{h} = -(\mathbf{C}\mathbf{v})^\wedge \mathbf{1}_i. \end{aligned} \quad (7.173)$$

Stacking the three directional derivatives together, we have

$$\frac{\partial(\mathbf{C}\mathbf{v})}{\partial\psi} = -(\mathbf{C}\mathbf{v})^\wedge, \quad (7.174)$$

¹⁸ A right-hand version is also possible.

¹⁹ This is sometimes called a (left) *Lie derivative*.

**Figure 7.3**

During optimization, we keep our nominal rotation, \mathbf{C}_{op} , in the Lie group and consider a perturbation, ψ , to take place in the Lie algebra, which is locally the tangent space of the group.

which is the same as our previous expression but without the \mathbf{J} .

An even simpler way to think about optimization is to skip the derivatives altogether and think in terms of perturbations. Choose a perturbation scheme,

$$\mathbf{C} = \exp(\psi^\wedge) \mathbf{C}_{\text{op}}, \quad (7.175)$$

where ψ is a small perturbation applied to an initial guess, \mathbf{C}_{op} . When we take the product of the rotation and a point, \mathbf{v} , we can approximate the expression as follows:

$$\mathbf{C}\mathbf{v} = \exp(\psi^\wedge) \mathbf{C}_{\text{op}}\mathbf{v} \approx \mathbf{C}_{\text{op}}\mathbf{v} - (\mathbf{C}_{\text{op}}\mathbf{v})^\wedge \psi. \quad (7.176)$$

This is depicted graphically in Figure 7.3. Inserting this perturbation scheme into the function to be optimized, we have

$$\begin{aligned} u(\mathbf{C}\mathbf{v}) &= u(\exp(\psi^\wedge) \mathbf{C}_{\text{op}}\mathbf{v}) \approx u((\mathbf{1} + \psi^\wedge) \mathbf{C}_{\text{op}}\mathbf{v}) \\ &\approx u(\mathbf{C}_{\text{op}}\mathbf{v}) - \underbrace{\frac{\partial u}{\partial \mathbf{x}} \Big|_{\mathbf{x}=\mathbf{C}_{\text{op}}\mathbf{v}} (\mathbf{C}_{\text{op}}\mathbf{v})^\wedge \psi}_{\delta^T} = u(\mathbf{C}_{\text{op}}\mathbf{v}) + \boldsymbol{\delta}^T \psi. \end{aligned} \quad (7.177)$$

Then pick a perturbation to decrease the function. For example, gradient descent suggests we would like to pick

$$\psi = -\alpha \mathbf{D} \boldsymbol{\delta}, \quad (7.178)$$

with $\alpha > 0$ a small step size and $\mathbf{D} > 0$ any positive-definite matrix. Then apply the perturbation within the scheme to update the rotation,

$$\mathbf{C}_{\text{op}} \leftarrow \exp(-\alpha \mathbf{D} \boldsymbol{\delta}^\wedge) \mathbf{C}_{\text{op}}, \quad (7.179)$$

and iterate to convergence. Our scheme guarantees $\mathbf{C}_{\text{op}} \in SO(3)$ at each iteration.

The perturbation idea generalizes to more interesting optimization schemes than basic gradient descent, which can be quite slow. Consider the alternate derivation of the Gauss-Newton optimization method

from Section 4.3.1. Suppose we have a general nonlinear, quadratic cost function of a rotation of the form,

$$J(\mathbf{C}) = \frac{1}{2} \sum_m (u_m(\mathbf{C}\mathbf{v}_m))^2, \quad (7.180)$$

where $u_m(\cdot)$ are scalar nonlinear functions and $\mathbf{v}_m \in \mathbb{R}^3$ are three-dimensional points. We begin with an initial guess for the optimal rotation, $\mathbf{C}_{\text{op}} \in SO(3)$, and then perturb this (on the left) according to

$$\mathbf{C} = \exp(\psi^\wedge) \mathbf{C}_{\text{op}}, \quad (7.181)$$

where ψ is the perturbation. We then apply our perturbation scheme inside each $u_m(\cdot)$ so that

$$\begin{aligned} u_m(\mathbf{C}\mathbf{v}_m) &= u_m(\exp(\psi^\wedge)\mathbf{C}_{\text{op}}\mathbf{v}_m) \approx u_m((\mathbf{1} + \psi^\wedge)\mathbf{C}_{\text{op}}\mathbf{v}_m) \\ &\approx \underbrace{u_m(\mathbf{C}_{\text{op}}\mathbf{v}_m)}_{\beta_m} - \underbrace{\frac{\partial u_m}{\partial \mathbf{x}} \Big|_{\mathbf{x}=\mathbf{C}_{\text{op}}\mathbf{v}_m} (\mathbf{C}_{\text{op}}\mathbf{v}_m)^\wedge \psi}_{\delta_m^T}, \end{aligned} \quad (7.182)$$

is a linearized version of $u_m(\cdot)$ in terms of our perturbation, ψ . Inserting this back into our cost function, we have

$$J(\mathbf{C}) \approx \frac{1}{2} \sum_m \left(\delta_m^T \psi + \beta_m \right)^2, \quad (7.183)$$

which is exactly quadratic in ψ . Taking the derivative of J with respect to ψ , we have

$$\frac{\partial J}{\partial \psi^T} = \sum_m \delta_m \left(\delta_m^T \psi + \beta_m \right). \quad (7.184)$$

We can set the derivative to zero to find the optimal perturbation, ψ^* , that minimizes J :

$$\left(\sum_m \delta_m \delta_m^T \right) \psi^* = - \sum_m \beta_m \delta_m. \quad (7.185)$$

This is a linear system of equations, which we can solve for ψ^* . We then apply this optimal perturbation to our initial guess, according to our perturbation scheme:

$$\mathbf{C}_{\text{op}} \leftarrow \exp(\psi^{*\wedge}) \mathbf{C}_{\text{op}}, \quad (7.186)$$

which ensures that at each iteration, we have $\mathbf{C}_{\text{op}} \in SO(3)$. We iterate to convergence and then output $\mathbf{C}^* = \mathbf{C}_{\text{op}}$ at the final iteration as our optimized rotation. This is exactly the Gauss-Newton algorithm, but adapted to work with the matrix Lie group, $SO(3)$, by exploiting the surjective-only property of the exponential map to define an appropriate perturbation scheme.

Poses

The same concepts can also be applied to poses. The Jacobian of a transformed point with respect to the Lie algebra vector representing the transformation is

$$\frac{\partial(\mathbf{T}\mathbf{p})}{\partial\xi} = (\mathbf{T}\mathbf{p})^\odot \mathcal{J}, \quad (7.187)$$

where $\mathbf{T} = \exp(\xi^\wedge) \in SE(3)$ and $\mathbf{p} \in \mathbb{R}^4$ is some arbitrary three-dimensional point, expressed in homogeneous coordinates.

However, if we perturb the transformation matrix on the left,

$$\mathbf{T} \leftarrow \exp(\epsilon^\wedge) \mathbf{T}, \quad (7.188)$$

then the Jacobian with respect to this perturbation (i.e., the (left) Lie derivative) is simply

$$\frac{\partial(\mathbf{T}\mathbf{p})}{\partial\epsilon} = (\mathbf{T}\mathbf{p})^\odot, \quad (7.189)$$

which removes the need to calculate the \mathcal{J} matrix.

Finally, for optimization, suppose we have a general nonlinear, quadratic cost function of a transformation of the form

$$J(\mathbf{T}) = \frac{1}{2} \sum_m (u_m(\mathbf{T}\mathbf{p}_m))^2, \quad (7.190)$$

where $u_m(\cdot)$ are nonlinear functions and $\mathbf{p}_m \in \mathbb{R}^4$ are three-dimensional points expressed in homogeneous coordinates. We begin with an initial guess for the optimal transformation, $\mathbf{T}_{\text{op}} \in SE(3)$, and then perturb this (on the left) according to

$$\mathbf{T} = \exp(\epsilon^\wedge) \mathbf{T}_{\text{op}}, \quad (7.191)$$

where ϵ is the perturbation. We then apply our perturbation scheme inside each $u_m(\cdot)$ so that

$$\begin{aligned} u_m(\mathbf{T}\mathbf{p}_m) &= u_m(\exp(\epsilon^\wedge)\mathbf{T}_{\text{op}}\mathbf{p}_m) \approx u_m((\mathbf{1} + \epsilon^\wedge)\mathbf{T}_{\text{op}}\mathbf{p}_m) \\ &\approx \underbrace{u_m(\mathbf{T}_{\text{op}}\mathbf{p}_m)}_{\beta_m} + \underbrace{\frac{\partial u_m}{\partial \mathbf{x}} \Big|_{\mathbf{x}=\mathbf{T}_{\text{op}}\mathbf{p}_m} (\mathbf{T}_{\text{op}}\mathbf{p}_m)^\odot \epsilon}_{\delta_m^T} \quad (7.192) \end{aligned}$$

is a linearized version of $u_m(\cdot)$ in terms of our perturbation, ϵ . Inserting this back into our cost function, we have

$$J(\mathbf{T}) = \frac{1}{2} \sum_m (\delta_m^T \epsilon + \beta_m)^2, \quad (7.193)$$

which is exactly quadratic in ϵ . Taking the derivative of J with respect to ϵ , we have

$$\frac{\partial J}{\partial \epsilon^T} = \sum_m \delta_m (\delta_m^T \epsilon + \beta_m). \quad (7.194)$$

SO(3) Identities and Approximations

Lie Algebra

Lie Group

(left) Jacobian

$$\begin{aligned}
\mathbf{u}^\wedge &= \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}^\wedge = \begin{bmatrix} 0 & -u_3 & u_2 \\ u_3 & 0 & -u_1 \\ -u_2 & u_1 & 0 \end{bmatrix} \\
(\alpha \mathbf{u} + \beta \mathbf{v})^\wedge &\equiv \alpha \mathbf{u}^\wedge + \beta \mathbf{v}^\wedge \\
\mathbf{u}^{\wedge_T} &\equiv -\mathbf{u}^\wedge \\
\mathbf{u}^\wedge \mathbf{v} &\equiv -\mathbf{v}^\wedge \mathbf{u} \\
\mathbf{u}^\wedge \mathbf{u} &\equiv \mathbf{0} \\
(\mathbf{W}\mathbf{u})^\wedge &\equiv \mathbf{u}^\wedge (\text{tr}(\mathbf{W})\mathbf{1} - \mathbf{W}) - \mathbf{W}^T \mathbf{u}^\wedge \\
\mathbf{u}^\wedge \mathbf{v}^\wedge &\equiv -(\mathbf{u}^T \mathbf{v})\mathbf{1} + \mathbf{v}\mathbf{u}^T \\
\mathbf{u}^\wedge \mathbf{W} \mathbf{v}^\wedge &\equiv -(-\text{tr}(\mathbf{v}\mathbf{u}^T)\mathbf{1} + \mathbf{v}\mathbf{u}^T) \\
&\times (-\text{tr}(\mathbf{W})\mathbf{1} + \mathbf{W}^T) \\
&+ \text{tr}(\mathbf{W}^T \mathbf{v}\mathbf{u}^T)\mathbf{1} - \mathbf{W}^T \mathbf{v}\mathbf{u}^T \\
\mathbf{u}^\wedge \mathbf{v}^\wedge \mathbf{u}^\wedge &\equiv \mathbf{u}^\wedge \mathbf{u}^\wedge \mathbf{v}^\wedge + \mathbf{v}^\wedge \mathbf{u}^\wedge \mathbf{u}^\wedge + (\mathbf{u}^T \mathbf{u}) \mathbf{v}^\wedge \\
(\mathbf{u}^\wedge)^3 &+ (\mathbf{u}^T \mathbf{u}) \mathbf{u}^\wedge \equiv 0 \\
\mathbf{u}^\wedge \mathbf{v}^\wedge \mathbf{v}^\wedge &+ \mathbf{v}^\wedge \mathbf{v}^\wedge \mathbf{u}^\wedge + 2(\mathbf{v}^T \mathbf{v}) \mathbf{u}^\wedge \equiv (\mathbf{v}^\wedge \mathbf{u}^\wedge \mathbf{v})^\wedge \\
[\mathbf{u}^\wedge, \mathbf{v}^\wedge] &\equiv \mathbf{u}^\wedge \mathbf{v}^\wedge - \mathbf{v}^\wedge \mathbf{u}^\wedge \equiv (\mathbf{u}^\wedge \mathbf{v})^\wedge \\
[\mathbf{u}^\wedge, [\mathbf{u}^\wedge, \dots [\mathbf{u}^\wedge, \mathbf{v}^\wedge] \dots]] &\equiv \underbrace{((\mathbf{u}^\wedge)^n \mathbf{v})^\wedge}_n
\end{aligned}$$

$$\begin{aligned}
\mathbf{C} &= \exp(\phi^\wedge) \equiv \sum_{n=0}^{\infty} \frac{1}{n!} (\phi^\wedge)^n \\
&\equiv \cos \phi \mathbf{1} + (1 - \cos \phi) \mathbf{a} \mathbf{a}^T + \sin \phi \mathbf{a}^\wedge \\
\mathbf{C}^{-1} &\equiv \mathbf{C}^T \equiv \sum_{n=0}^{\infty} \frac{1}{n!} (-\phi^\wedge)^n \approx \mathbf{1} - \phi^\wedge \\
\phi &= \phi \mathbf{a} \\
\mathbf{a}^T \mathbf{a} &\equiv 1 \\
\mathbf{C}^T \mathbf{C} &\equiv \mathbf{1} \equiv \mathbf{C} \mathbf{C}^T \\
\text{tr}(\mathbf{C}) &\equiv 2 \cos \phi + 1 \\
\det(\mathbf{C}) &\equiv 1 \\
\mathbf{C} \mathbf{a} &\equiv \mathbf{a} \\
\mathbf{C} \phi &= \phi \\
\mathbf{C} \mathbf{a}^\wedge &\equiv \mathbf{a}^\wedge \mathbf{C} \\
\mathbf{C} \phi^\wedge &\equiv \phi^\wedge \mathbf{C} \\
(\mathbf{C} \mathbf{u})^\wedge &\equiv \mathbf{C} \mathbf{u}^\wedge \mathbf{C}^T \\
\exp((\mathbf{C} \mathbf{u})^\wedge) &\approx \exp((\mathbf{J} \delta \phi)^\wedge) \exp(\phi^\wedge) \\
\mathbf{C} &\equiv \mathbf{1} + \phi^\wedge \mathbf{J} \\
\mathbf{J}(\phi) &\equiv \mathbf{C} \mathbf{J}(-\phi)
\end{aligned}$$

$$\begin{aligned}
(\mathbf{A}^\alpha \mathbf{C})^\alpha &\approx (1 + (\mathbf{A}(\alpha, \phi) \delta \phi)^\wedge) \mathbf{C}^\alpha \\
\mathbf{A}(\alpha, \phi) &= \alpha \mathbf{J}(\alpha \phi) \mathbf{J}(\phi)^{-1} = \sum_{n=0}^{\infty} \frac{F_n(\alpha)}{n!} (\phi^\wedge)^n
\end{aligned}$$

$\alpha, \beta \in \mathbb{R}$, $\mathbf{u}, \mathbf{v}, \phi, \delta \phi \in \mathbb{R}^3$, $\mathbf{W}, \mathbf{A}, \mathbf{J} \in \mathbb{R}^{3 \times 3}$, $\mathbf{C} \in SO(3)$

Lie Algebra

Lie Group

(left) Jacobian

$$\begin{aligned}
& \mathbf{x}^\wedge = \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix}^\wedge = \begin{bmatrix} \mathbf{v}^\wedge & \mathbf{u} \end{bmatrix} \\
& \mathbf{x}^\wedge = \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix}^\wedge = \begin{bmatrix} \mathbf{v}^\wedge & \mathbf{u}^\wedge \end{bmatrix} \\
& (\alpha \mathbf{x} + \beta \mathbf{y})^\wedge \equiv \alpha \mathbf{x}^\wedge + \beta \mathbf{y}^\wedge \\
& (\alpha \mathbf{x} + \beta \mathbf{y})_\wedge \equiv \alpha \mathbf{x}^\wedge + \beta \mathbf{y}^\wedge \\
& \mathbf{x}^\wedge \mathbf{y} \equiv -\mathbf{y}^\wedge \mathbf{x} \\
& \mathbf{x}^\wedge \mathbf{x} \equiv \mathbf{0} \\
& (\mathbf{x}^\wedge)^4 + (\mathbf{v}^T \mathbf{v})(\mathbf{x}^\wedge)^2 \equiv 0 \\
& (\mathbf{x}^\wedge)^5 + 2(\mathbf{v}^T \mathbf{v})(\mathbf{x}^\wedge)^3 + (\mathbf{v}^T \mathbf{v})^2(\mathbf{x}^\wedge) \equiv 0 \\
& [\mathbf{x}^\wedge, \mathbf{y}^\wedge] \equiv \mathbf{x}^\wedge \mathbf{y}^\wedge - \mathbf{y}^\wedge \mathbf{x}^\wedge \equiv (\mathbf{x}^\wedge \mathbf{y})^\wedge \\
& [\mathbf{x}^\wedge, \mathbf{y}^\wedge] \equiv \mathbf{x}^\wedge \mathbf{y}^\wedge - \mathbf{y}^\wedge \mathbf{x}^\wedge \equiv (\mathbf{x}^\wedge \mathbf{y})^\wedge \\
& \underbrace{[\mathbf{x}^\wedge, [\mathbf{x}^\wedge, \dots, [\mathbf{x}^\wedge, \mathbf{y}^\wedge] \dots]]}_{n} \equiv ((\mathbf{x}^\wedge)^n \mathbf{y})^\wedge \\
& \mathbf{p}^\odot = \begin{bmatrix} \boldsymbol{\varepsilon} \\ \eta \end{bmatrix}^\odot = \begin{bmatrix} \eta \mathbf{1} & -\boldsymbol{\varepsilon}^\wedge \\ \mathbf{0}^T & \mathbf{0}^T \end{bmatrix} \\
& \mathbf{p}^\odot = \begin{bmatrix} \boldsymbol{\varepsilon} \\ \eta \end{bmatrix}^\odot = \begin{bmatrix} \mathbf{0} & \boldsymbol{\varepsilon} \\ -\boldsymbol{\varepsilon}^\wedge & \mathbf{0} \end{bmatrix} \\
& \mathbf{x}^\wedge \mathbf{p} \equiv \mathbf{p}^\odot \mathbf{x} \\
& \mathbf{p}^T \mathbf{x}^\wedge \equiv \mathbf{x}^T \mathbf{p}^\odot
\end{aligned}$$

$$\begin{aligned}
& \boldsymbol{\xi} = \begin{bmatrix} \rho \\ \boldsymbol{\phi} \end{bmatrix} \\
& \mathbf{T} = \exp(\boldsymbol{\xi}^\wedge) \equiv \sum_{n=0}^{\infty} \frac{1}{n!} (\boldsymbol{\xi}^\wedge)^n \\
& \equiv \mathbf{1} + \boldsymbol{\xi}^\wedge + \left(\frac{1-\cos\phi}{\phi^2} \right) (\boldsymbol{\xi}^\wedge)^2 + \left(\frac{\phi - \sin\phi}{\phi^3} \right) (\boldsymbol{\xi}^\wedge)^3 \\
& \approx \mathbf{1} + \boldsymbol{\xi}^\wedge \\
& \mathbf{T} \equiv \begin{bmatrix} \mathbf{C} & \mathbf{J}\boldsymbol{\rho} \\ \mathbf{0}^T & 1 \end{bmatrix} \\
& \boldsymbol{\xi}^\wedge \equiv \text{ad}(\boldsymbol{\xi}^\wedge) \\
& \mathcal{T} = \exp(\boldsymbol{\xi}^\wedge) \equiv \sum_{n=0}^{\infty} \frac{1}{n!} (\boldsymbol{\xi}^\wedge)^n \\
& \equiv \mathbf{1} + \left(\frac{3 \sin\phi - \phi \cos\phi}{2\phi} \right) \boldsymbol{\xi}^\wedge + \left(\frac{4 \phi - 5 \sin\phi + \phi \cos\phi}{2\phi^3} \right) (\boldsymbol{\xi}^\wedge)^2 \\
& + \left(\frac{2 - \phi \sin\phi - 2 \cos\phi}{2\phi^4} \right) (\boldsymbol{\xi}^\wedge)^3 + \left(\frac{2\phi - 3 \sin\phi + \phi \cos\phi}{2\phi^5} \right) (\boldsymbol{\xi}^\wedge)^4 \\
& \approx \mathbf{1} + \frac{1}{2} \boldsymbol{\xi}^\wedge \\
& \mathcal{T} \equiv \begin{bmatrix} \mathbf{J} & \mathbf{Q} \\ \mathbf{0} & \mathbf{J} \end{bmatrix} \\
& \mathcal{T}^{-1} \equiv \sum_{n=0}^{\infty} \frac{B_n}{n!} (\boldsymbol{\xi}^\wedge)^n \approx \mathbf{1} - \frac{1}{2} \boldsymbol{\xi}^\wedge \\
& \mathcal{T}^{-1} \equiv \begin{bmatrix} \mathbf{0} & \mathbf{J} \\ \mathbf{J}^{-1} & -\mathbf{J}^{-1} \mathbf{Q} \mathbf{J}^{-1} \end{bmatrix} \\
& \mathbf{Q} = \sum_{m=0}^{\infty} \sum_{n=0}^{\infty} \frac{1}{(n+m+2)!} (\boldsymbol{\phi}^\wedge)^n \boldsymbol{\rho}^\wedge (\boldsymbol{\phi}^\wedge)^m \\
& \equiv \frac{1}{2} \boldsymbol{\rho}^\wedge + \left(\frac{\phi - \sin\phi}{\phi^3} \right) (\boldsymbol{\phi}^\wedge \boldsymbol{\rho}^\wedge + \boldsymbol{\rho}^\wedge \boldsymbol{\phi}^\wedge + \boldsymbol{\phi}^\wedge \boldsymbol{\rho}^\wedge \boldsymbol{\phi}^\wedge) \\
& + \left(\frac{\phi^2 + 2 \cos\phi - 2}{2\phi^4} \right) (\boldsymbol{\phi}^\wedge \boldsymbol{\phi}^\wedge \boldsymbol{\rho}^\wedge + \boldsymbol{\rho}^\wedge \boldsymbol{\phi}^\wedge \boldsymbol{\phi}^\wedge - 3\boldsymbol{\phi}^\wedge \boldsymbol{\rho}^\wedge \boldsymbol{\phi}^\wedge) \\
& + \left(\frac{2\phi - 3 \sin\phi + \phi \cos\phi}{2\phi^5} \right) (\boldsymbol{\phi}^\wedge \boldsymbol{\rho}^\wedge \boldsymbol{\phi}^\wedge \boldsymbol{\phi}^\wedge + \boldsymbol{\phi}^\wedge \boldsymbol{\phi}^\wedge \boldsymbol{\rho}^\wedge \boldsymbol{\phi}^\wedge) \\
& \exp((\boldsymbol{\xi} + \delta\boldsymbol{\xi})^\wedge) \approx \exp((\mathcal{J}(\delta\boldsymbol{\xi}))^\wedge) \exp(\boldsymbol{\xi}^\wedge) \\
& \exp((\boldsymbol{\xi} + \delta\boldsymbol{\xi})^\wedge) \approx \exp((\mathcal{J}(\delta\boldsymbol{\xi}))^\wedge) \exp(\boldsymbol{\xi}^\wedge) \\
& \mathcal{T} \equiv \mathbf{1} + \boldsymbol{\xi}^\wedge \mathcal{J} \\
& \mathcal{J} \boldsymbol{\xi}^\wedge \equiv \mathcal{T} \mathcal{J}(-\boldsymbol{\xi}) \\
& \mathcal{J}(\boldsymbol{\xi}) \equiv \mathcal{T} \mathcal{J}(-\boldsymbol{\xi}) \\
& \mathbf{T} \boldsymbol{\xi}^\wedge \equiv \boldsymbol{\xi}^\wedge \mathbf{T}, \quad \mathcal{T} \boldsymbol{\xi}^\wedge \equiv \boldsymbol{\xi}^\wedge \mathcal{T} \\
& (\mathcal{T} \mathbf{x})^\wedge \equiv \mathbf{T} \mathbf{x}^\wedge \mathbf{T}^{-1}, \quad (\mathcal{T} \mathbf{x})^\wedge \equiv \mathcal{T} \mathbf{x}^\wedge \mathcal{T}^{-1} \\
& \exp((\mathcal{T} \mathbf{x})^\wedge) \equiv \mathbf{T} \exp(\mathbf{x}^\wedge) \mathbf{T}^{-1} \\
& \exp((\mathcal{T} \mathbf{x})^\wedge) \equiv \mathcal{T} \exp(\mathbf{x}^\wedge) \mathcal{T}^{-1} \\
& (\mathbf{T} \mathbf{p})^\odot \equiv \mathbf{T} \mathbf{p}^\odot \mathcal{T}^{-1} \\
& (\mathbf{T} \mathbf{p})^\odot \equiv \mathcal{T}^{-T} \mathbf{p}^{\odot T} \mathbf{p}^\odot \mathcal{T}^{-1}
\end{aligned}$$

$\alpha, \beta \in \mathbb{R}$, $\mathbf{u}, \mathbf{v}, \boldsymbol{\phi}, \delta\boldsymbol{\phi} \in \mathbb{R}^3$, $\mathbf{p} \in \mathbb{R}^4$, $\mathbf{x}, \mathbf{y}, \boldsymbol{\xi}, \delta\boldsymbol{\xi} \in \mathbb{R}^6$, $\mathbf{C} \in SO(3)$, $\mathbf{J}, \mathbf{Q} \in \mathbb{R}^{3 \times 3}$, $\mathbf{T}, \mathbf{T}_1, \mathbf{T}_2 \in SE(3)$, $\mathcal{T} \in \text{Ad}(SE(3))$, $\mathcal{J}, \mathcal{A} \in \mathbb{R}^{6 \times 6}$

We can set the derivative to zero to find the optimal perturbation, ϵ^* , that minimizes J :

$$\left(\sum_m \boldsymbol{\delta}_m \boldsymbol{\delta}_m^T \right) \boldsymbol{\epsilon}^* = - \sum_m \beta_m \boldsymbol{\delta}_m. \quad (7.195)$$

This is a linear system of equations, which we can solve for $\boldsymbol{\epsilon}^*$. We then apply this optimal perturbation to our initial guess, according to our perturbation scheme,

$$\mathbf{T}_{\text{op}} \leftarrow \exp(\boldsymbol{\epsilon}^{*\wedge}) \mathbf{T}_{\text{op}}, \quad (7.196)$$

which ensures that at each iteration, we have $\mathbf{T}_{\text{op}} \in SE(3)$. We iterate to convergence and then output $\mathbf{T}^* = \mathbf{T}_{\text{op}}$ at the final iteration as the optimal pose. This is exactly the Gauss-Newton algorithm, but adapted to work with the matrix Lie group, $SE(3)$, by exploiting the surjective-only property of the exponential map to define an appropriate perturbation scheme.

Gauss-Newton Discussion

This approach to Gauss-Newton optimization for our matrix Lie groups where we use a customized perturbation scheme has three key properties:

- (i) We are storing our rotation or pose in a singularity-free format,
- (ii) At each iteration we are performing unconstrained optimization,
- (iii) Our manipulations occur at the matrix level so that we do not need to worry about taking the derivatives of a bunch of scalar trigonometric functions, which can easily lead to mistakes.

This makes implementation quite straightforward. We can also easily incorporate both of the practical patches to Gauss-Newton that were outlined in Section 4.3.1 (a line search and Levenberg-Marquardt) as well as the ideas from robust estimation described in 5.3.2.

7.1.10 Identities

We have seen many identities and expressions in this section related to our matrix Lie groups, $SO(3)$ and $SE(3)$. The previous two pages summarize these. The first page provides identities for $SO(3)$ and the second for $SE(3)$.

7.2 Kinematics

We have seen how the geometry of a Lie group works. The next step is to allow the geometry to change over time. We will work out the *kinematics* associated with our two Lie groups, $SO(3)$ and $SE(3)$.

7.2.1 Rotations

We have already seen the kinematics of rotations in the previous chapter, but this was before we introduced Lie groups.

Lie Group

We know that a rotation matrix can be written as

$$\mathbf{C} = \exp(\boldsymbol{\phi}^\wedge), \quad (7.197)$$

where $\mathbf{C} \in SO(3)$ and $\boldsymbol{\phi} = \phi \mathbf{a} \in \mathbb{R}^3$. The rotational kinematic equation relating angular velocity, $\boldsymbol{\omega}$, to rotation (i.e., Poisson's equation) is given by²⁰

$$\dot{\mathbf{C}} = \boldsymbol{\omega}^\wedge \mathbf{C} \quad \text{or} \quad \boldsymbol{\omega}^\wedge = \dot{\mathbf{C}} \mathbf{C}^T. \quad (7.198)$$

We will refer to this as kinematics of the Lie group; these equations are singularity-free since they are in terms of \mathbf{C} , but have the constraint that $\mathbf{C}\mathbf{C}^T = \mathbf{1}$. Owing to the surjective-only property of the exponential map from $\mathfrak{so}(3)$ to $SO(3)$, we can also work out the kinematics in terms of the Lie algebra.

Lie Algebra

To see the equivalent kinematics in terms of the Lie algebra, we need to differentiate \mathbf{C} :

$$\dot{\mathbf{C}} = \frac{d}{dt} \exp(\boldsymbol{\phi}^\wedge) = \int_0^1 \exp(\alpha \boldsymbol{\phi}^\wedge) \dot{\boldsymbol{\phi}}^\wedge \exp((1-\alpha) \boldsymbol{\phi}^\wedge) d\alpha, \quad (7.199)$$

where the last relationship comes from the general expression for the time derivative of the matrix exponential:

$$\frac{d}{dt} \exp(\mathbf{A}(t)) = \int_0^1 \exp(\alpha \mathbf{A}(t)) \frac{d\mathbf{A}(t)}{dt} \exp((1-\alpha) \mathbf{A}(t)) d\alpha. \quad (7.200)$$

²⁰ Compared to (6.45) in our earlier development, this $\boldsymbol{\omega}$ is opposite in sign. This is because we have adopted the robotics convention described in Section 6.3.2 for the angle of rotation, and this leads to the form in (7.197); this in turn means we must use the angular velocity associated with that angle, and this is opposite in sign to the one we discussed earlier.

From (7.199) we can rearrange to have

$$\begin{aligned}\dot{\mathbf{C}}\mathbf{C}^T &= \int_0^1 \mathbf{C}^\alpha \dot{\phi}^\wedge \mathbf{C}^{-\alpha} d\alpha = \int_0^1 (\mathbf{C}^\alpha \dot{\phi})^\wedge d\alpha \\ &= \left(\int_0^1 \mathbf{C}^\alpha d\alpha \dot{\phi} \right)^\wedge = (\mathbf{J} \dot{\phi})^\wedge,\end{aligned}\quad (7.201)$$

where $\mathbf{J} = \int_0^1 \mathbf{C}^\alpha d\alpha$ is the (left) Jacobian for $SO(3)$ that we saw earlier. Comparing (7.198) and (7.201) we have the pleasing result that

$$\boldsymbol{\omega} = \mathbf{J} \dot{\phi}, \quad (7.202)$$

or

$$\dot{\phi} = \mathbf{J}^{-1} \boldsymbol{\omega}, \quad (7.203)$$

which is an equivalent expression for the kinematics but in terms of the Lie algebra. Note that \mathbf{J}^{-1} does not exist at $|\phi| = 2\pi m$, where m is a non-zero integer, due to singularities of the 3×1 representation of rotation; the good news is that we no longer have constraints to worry about.

Numerical Integration

Because ϕ has no constraints, we can use any numerical method we like to integrate (7.203). The same is not true if we want to integrate (7.198) directly, since we must enforce the constraint that $\mathbf{C}\mathbf{C}^T = \mathbf{1}$. There are a few simple strategies we can use to do this.

One approach is to assume that $\boldsymbol{\omega}$ is piecewise constant. Suppose $\boldsymbol{\omega}$ is constant between two times, t_1 and t_2 . In this case, (7.198) is a linear, time-invariant, ordinary differential equation, and we know the solution will be of the form

$$\mathbf{C}(t_2) = \underbrace{\exp((t_2 - t_1)\boldsymbol{\omega}^\wedge)}_{\mathbf{C}_{21} \in SO(3)} \mathbf{C}(t_1), \quad (7.204)$$

where we note that \mathbf{C}_{21} is in fact in the correct form to be a rotation matrix. Let the rotation vector be

$$\boldsymbol{\phi} = \phi \mathbf{a} = (t_2 - t_1) \boldsymbol{\omega}, \quad (7.205)$$

with angle, $\phi = |\boldsymbol{\phi}|$, and axis, $\mathbf{a} = \boldsymbol{\phi}/\phi$. Then construct the rotation matrix through our usual closed-form expression:

$$\mathbf{C}_{21} = \cos \phi \mathbf{1} + (1 - \cos \phi) \mathbf{a} \mathbf{a}^T + \sin \phi \mathbf{a}^\wedge. \quad (7.206)$$

The update then proceeds as

$$\mathbf{C}(t_2) = \mathbf{C}_{21} \mathbf{C}(t_1), \quad (7.207)$$

which mathematically guarantees that $\mathbf{C}(t_2)$ will be in $SO(3)$ since

$\mathbf{C}_{21}, \mathbf{C}(t_1) \in SO(3)$. Repeating this over and over for many small time intervals allows us to integrate the equation numerically.

However, even if we do follow an integration approach (such as the one above) that claims to keep the computed rotation in $SO(3)$, small numerical errors may eventually cause the result to depart $SO(3)$ through violation of the orthogonality constraint. A common solution is to periodically ‘project’ the computed rotation, $\mathbf{C} \notin SO(3)$, back onto $SO(3)$. In other words, we can try to find the rotation matrix, $\mathbf{R} \in SO(3)$, that is closest to \mathbf{C} in some sense. We do this by solving the following optimization problem (Green, 1952):

$$\arg \max_{\mathbf{R}} J(\mathbf{R}), \quad J(\mathbf{R}) = \text{tr} (\mathbf{CR}^T) - \underbrace{\frac{1}{2} \sum_{i=1}^3 \sum_{j=1}^3 \lambda_{ij} (\mathbf{r}_i^T \mathbf{r}_j - \delta_{ij})}_{\text{Lagrange multiplier terms}}, \quad (7.208)$$

where the Lagrange multiplier terms are necessary to enforce the $\mathbf{RR}^T = \mathbf{1}$ constraint. Note that δ_{ij} is the Kronecker delta and

$$\mathbf{R}^T = [\mathbf{r}_1 \quad \mathbf{r}_2 \quad \mathbf{r}_3], \quad \mathbf{C}^T = [\mathbf{c}_1 \quad \mathbf{c}_2 \quad \mathbf{c}_3]. \quad (7.209)$$

We also note that

$$\text{tr} (\mathbf{CR}^T) = \mathbf{r}_1^T \mathbf{c}_1 + \mathbf{r}_2^T \mathbf{c}_2 + \mathbf{r}_3^T \mathbf{c}_3. \quad (7.210)$$

We then take the derivative of J with respect to the three rows of \mathbf{R} , revealing

$$\frac{\partial J}{\partial \mathbf{r}_i^T} = \mathbf{c}_i - \sum_{j=1}^3 \lambda_{ij} \mathbf{r}_j, \quad \forall i = 1 \dots 3. \quad (7.211)$$

Setting this to zero, $\forall i = 1 \dots 3$, we have that

$$\underbrace{[\mathbf{r}_1 \quad \mathbf{r}_2 \quad \mathbf{r}_3]}_{\mathbf{R}^T} \underbrace{\begin{bmatrix} \lambda_{11} & \lambda_{12} & \lambda_{13} \\ \lambda_{21} & \lambda_{22} & \lambda_{23} \\ \lambda_{31} & \lambda_{32} & \lambda_{33} \end{bmatrix}}_{\boldsymbol{\Lambda}} = \underbrace{[\mathbf{c}_1 \quad \mathbf{c}_2 \quad \mathbf{c}_3]}_{\mathbf{C}^T}. \quad (7.212)$$

Note, however, that $\boldsymbol{\Lambda}$ can be assumed to be symmetric owing to the symmetry of the Lagrange multiplier terms. Thus, what we know so far is that

$$\boldsymbol{\Lambda} \mathbf{R} = \mathbf{C}, \quad \boldsymbol{\Lambda} = \boldsymbol{\Lambda}^T, \quad \mathbf{R}^T \mathbf{R} = \mathbf{RR}^T = \mathbf{1}.$$

We can solve for $\boldsymbol{\Lambda}$ by noticing

$$\boldsymbol{\Lambda}^2 = \boldsymbol{\Lambda} \boldsymbol{\Lambda}^T = \boldsymbol{\Lambda} \underbrace{\mathbf{RR}^T}_{\mathbf{1}} \boldsymbol{\Lambda}^T = \mathbf{CC}^T \Rightarrow \boldsymbol{\Lambda} = (\mathbf{CC}^T)^{\frac{1}{2}},$$

with $(\cdot)^{\frac{1}{2}}$ indicating a matrix square-root. Finally,

$$\mathbf{R} = (\mathbf{CC}^T)^{-\frac{1}{2}} \mathbf{C},$$

which simply looks like we are ‘normalizing’ \mathbf{C} . Computing the projection whenever the orthogonality constraint is not satisfied (to within some threshold) and then overwriting the integrated value,

$$\mathbf{C} \leftarrow \mathbf{R}, \quad (7.213)$$

ensures that we do not stray too far from $SO(3)^{21}$.

7.2.2 Poses

There is an analogous approach to kinematics for $SE(3)$ that we will develop next.

Lie Group

We have seen that a transformation matrix can be written as

$$\mathbf{T} = \begin{bmatrix} \mathbf{C} & \mathbf{r} \\ \mathbf{0}^T & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{C} & \mathbf{J}\boldsymbol{\rho} \\ \mathbf{0}^T & 1 \end{bmatrix} = \exp(\boldsymbol{\xi}^\wedge), \quad (7.214)$$

where

$$\boldsymbol{\xi} = \begin{bmatrix} \boldsymbol{\rho} \\ \phi \end{bmatrix}.$$

Suppose the kinematics in terms of separated translation and rotation are given by

$$\dot{\mathbf{r}} = \boldsymbol{\omega}^\wedge \mathbf{r} + \boldsymbol{\nu}, \quad (7.215a)$$

$$\dot{\mathbf{C}} = \boldsymbol{\omega}^\wedge \mathbf{C}, \quad (7.215b)$$

where $\boldsymbol{\nu}$ and $\boldsymbol{\omega}$ are the translational and rotational velocities, respectively. Using transformation matrices, this can be written equivalently as

$$\dot{\mathbf{T}} = \boldsymbol{\varpi}^\wedge \mathbf{T} \quad \text{or} \quad \boldsymbol{\varpi}^\wedge = \dot{\mathbf{T}} \mathbf{T}^{-1}, \quad (7.216)$$

where

$$\boldsymbol{\varpi} = \begin{bmatrix} \boldsymbol{\nu} \\ \boldsymbol{\omega} \end{bmatrix},$$

is the *generalized velocity*²². Again, these equations are singularity-free but still have the constraint that $\mathbf{C}\mathbf{C}^T = \mathbf{1}$.

²¹ Technically, this matrix square-root approach only works under certain conditions. For some pathological \mathbf{C} matrices, it can produce an \mathbf{R} where $\det \mathbf{R} = -1$ instead of $\det \mathbf{R} = 1$, as desired. This is because we have not enforced the $\det \mathbf{R} = 1$ constraint in our optimization properly. A more rigorous method, based on singular-value decomposition and that handles the more difficult cases, is presented later in Section 8.1.3. A sufficient test to know whether this matrix square-root approach will work is to check that $\det \mathbf{C} > 0$ before applying it. This should almost always be true in real situations where our integration step is small. If it is not true, the detailed method in Section 8.1.3 should be used.

²² We can also write the kinematics equivalently in 6×6 format:

$$\dot{\mathcal{T}} = \boldsymbol{\varpi}^\wedge \mathcal{T}.$$

Lie Algebra

Again, we can find an equivalent set of kinematics in terms of the Lie algebra. As in the rotation case, we have that

$$\dot{\mathbf{T}} = \frac{d}{dt} \exp(\boldsymbol{\xi}^\wedge) = \int_0^1 \exp(\alpha \boldsymbol{\xi}^\wedge) \dot{\boldsymbol{\xi}}^\wedge \exp((1-\alpha)\boldsymbol{\xi}^\wedge) d\alpha, \quad (7.217)$$

or equivalently,

$$\begin{aligned} \dot{\mathbf{T}}\mathbf{T}^{-1} &= \int_0^1 \mathbf{T}^\alpha \dot{\boldsymbol{\xi}}^\wedge \mathbf{T}^{-\alpha} d\alpha = \int_0^1 (\mathcal{T}^\alpha \dot{\boldsymbol{\xi}})^\wedge d\alpha \\ &= \left(\left(\int_0^1 \mathcal{T}^\alpha d\alpha \right) \dot{\boldsymbol{\xi}} \right)^\wedge = (\mathcal{J}\dot{\boldsymbol{\xi}})^\wedge, \end{aligned} \quad (7.218)$$

where $\mathcal{J} = \int_0^1 \mathcal{T}^\alpha d\alpha$ is the (left) Jacobian for $SE(3)$. Comparing (7.216) and (7.218), we have that

$$\boldsymbol{\varpi} = \mathcal{J}\dot{\boldsymbol{\xi}}, \quad (7.219)$$

or

$$\dot{\boldsymbol{\xi}} = \mathcal{J}^{-1}\boldsymbol{\varpi}, \quad (7.220)$$

for our equivalent kinematics in terms of the Lie algebra. Again, these equations are now free of constraints.

Hybrid

There is, however, another way to propagate the kinematics by noting that the equation for $\dot{\mathbf{r}}$ is actually linear in the velocity. By combining the equations for $\dot{\mathbf{r}}$ and $\dot{\phi}$, we have

$$\begin{bmatrix} \dot{\mathbf{r}} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \mathbf{1} & -\mathbf{r}^\wedge \\ \mathbf{0} & \mathbf{J}^{-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\nu} \\ \boldsymbol{\omega} \end{bmatrix}, \quad (7.221)$$

which still has singularities at the singularities of \mathbf{J}^{-1} but no longer requires us to evaluate \mathbf{Q} and avoids the conversion $\mathbf{r} = \mathbf{J}\boldsymbol{\rho}$ after we integrate. This approach is also free of constraints. We can refer to this as a *hybrid* method, as the translation is kept in the usual space and the rotation is kept in the Lie algebra.

Numerical Integration

Similarly to the $SO(3)$ approach, we can integrate (7.220) without worrying about constraints, but integrating (7.216) takes a little more care.

Just as in the $SO(3)$ approach, we could assume that $\boldsymbol{\varpi}$ is piecewise constant. Suppose $\boldsymbol{\varpi}$ is constant between two times, t_1 and t_2 . In this

case, (7.216) is a linear, time-invariant, ordinary differential equation, and we know the solution will be of the form

$$\mathbf{T}(t_2) = \underbrace{\exp((t_2 - t_1)\boldsymbol{\varpi}^\wedge)}_{\mathbf{T}_{21} \in SE(3)} \mathbf{T}(t_1), \quad (7.222)$$

where we note that \mathbf{T}_{21} is in fact in the correct form to be a transformation matrix. Let

$$\boldsymbol{\xi} = \begin{bmatrix} \boldsymbol{\rho} \\ \phi \end{bmatrix} = (t_2 - t_1) \begin{bmatrix} \boldsymbol{\nu} \\ \boldsymbol{\omega} \end{bmatrix} = (t_2 - t_1) \boldsymbol{\varpi}, \quad (7.223)$$

with angle, $\phi = |\boldsymbol{\rho}|$, and axis, $\mathbf{a} = \boldsymbol{\rho}/\phi$. Then construct the rotation matrix through our usual closed-form expression:

$$\mathbf{C} = \cos \phi \mathbf{1} + (1 - \cos \phi) \mathbf{a} \mathbf{a}^T + \sin \phi \mathbf{a}^\wedge. \quad (7.224)$$

Build \mathbf{J} and calculate $\mathbf{r} = \mathbf{J}\boldsymbol{\rho}$. Assemble \mathbf{C} and \mathbf{r} into

$$\mathbf{T}_{21} = \begin{bmatrix} \mathbf{C} & \mathbf{r} \\ \mathbf{0}^T & 1 \end{bmatrix}. \quad (7.225)$$

The update then proceeds as

$$\mathbf{T}(t_2) = \mathbf{T}_{21} \mathbf{T}(t_1), \quad (7.226)$$

which mathematically guarantees that $\mathbf{T}(t_2)$ will be in $SE(3)$ since $\mathbf{T}_{21}, \mathbf{T}(t_1) \in SE(3)$. Repeating this over and over for many small time intervals allows us to integrate the equation numerically. We can also project the upper-left, rotation matrix part of \mathbf{T} back onto $SO(3)$ periodically²³, reset the lower left block to $\mathbf{0}^T$, and reset the lower-right block to 1, to ensure \mathbf{T} does not stray too far from $SE(3)$.

With Dynamics

We can augment our kinematic equation for pose with an equation for the translational/rotational dynamics (i.e., Newton's second law) as follows (D'Eleuterio, 1985):

$$\text{kinematics: } \dot{\mathbf{T}} = \boldsymbol{\varpi}^\wedge \mathbf{T}, \quad (7.227a)$$

$$\text{dynamics: } \dot{\boldsymbol{\varpi}} = -\mathbf{M}^{-1} \boldsymbol{\varpi}^{\wedge T} \mathbf{M} \boldsymbol{\varpi} + \mathbf{a}, \quad (7.227b)$$

where $\mathbf{T} \in SE(3)$ is the pose, $\boldsymbol{\varpi} \in \mathbb{R}^6$ is the generalized velocity (in the body frame), $\mathbf{a} \in \mathbb{R}^6$ is a generalized applied force (per mass, in the body frame), and $\mathbf{M} \in \mathbb{R}^{6 \times 6}$ is a generalized mass matrix of the form

$$\mathbf{M} = \begin{bmatrix} m\mathbf{1} & -m\mathbf{c}^\wedge \\ m\mathbf{c}^\wedge & \mathbf{I} \end{bmatrix}, \quad (7.228)$$

with m the mass, $\mathbf{c} \in \mathbb{R}^3$ the center of mass, and $\mathbf{I} \in \mathbb{R}^{3 \times 3}$ the inertia matrix, all in the body frame.

²³ See the numerical integration section on rotations, above, for the details.

7.2.3 Linearized Rotations

Lie Group

We can also perturb our kinematics about some nominal solution (i.e., linearize), both in the Lie group and the Lie algebra. We begin with the Lie group. Consider the following perturbed rotation matrix, $\mathbf{C}' \in SO(3)$:

$$\mathbf{C}' = \exp(\delta\phi^\wedge) \mathbf{C} \approx (\mathbf{1} + \delta\phi^\wedge) \mathbf{C}, \quad (7.229)$$

where $\mathbf{C} \in SO(3)$ is the nominal rotation matrix and $\delta\phi \in \mathbb{R}^3$ is a perturbation as a rotation vector. The perturbed kinematics equation, $\dot{\mathbf{C}}' = \boldsymbol{\omega}'^\wedge \mathbf{C}'$, becomes

$$\underbrace{\frac{d}{dt} ((\mathbf{1} + \delta\phi^\wedge) \mathbf{C})}_{\dot{\mathbf{C}}'} \approx \underbrace{(\boldsymbol{\omega} + \delta\boldsymbol{\omega})^\wedge}_{\boldsymbol{\omega}'} \underbrace{(\mathbf{1} + \delta\phi^\wedge) \mathbf{C}}_{\mathbf{C}'}, \quad (7.230)$$

after inserting our perturbation scheme. Dropping products of small terms, we can manipulate this into a pair of equations,

$$\text{nominal kinematics: } \dot{\mathbf{C}} = \boldsymbol{\omega}^\wedge \mathbf{C}, \quad (7.231a)$$

$$\text{perturbation kinematics: } \dot{\delta\phi} = \boldsymbol{\omega}^\wedge \delta\phi + \delta\boldsymbol{\omega}, \quad (7.231b)$$

which can be integrated separately and combined to provide the complete solution (approximately).

Lie Algebra

Perturbing the kinematics in the Lie algebra is more difficult but equivalent. In terms of quantities in the Lie algebra, we have

$$\boldsymbol{\phi}' = \boldsymbol{\phi} + \mathbf{J}(\boldsymbol{\phi})^{-1} \delta\boldsymbol{\phi}, \quad (7.232)$$

where $\boldsymbol{\phi}' = \ln(\mathbf{C}')^\vee$ is the perturbed rotation vector, $\boldsymbol{\phi} = \ln(\mathbf{C})^\vee$ the nominal rotation vector, and $\delta\boldsymbol{\phi}$ the same perturbation as in the Lie group case.

We start with the perturbed kinematics, $\dot{\boldsymbol{\phi}}' = \mathbf{J}(\boldsymbol{\phi}')^{-1} \boldsymbol{\omega}'$, and then inserting our perturbation scheme, we have

$$\underbrace{\frac{d}{dt} (\boldsymbol{\phi} + \mathbf{J}(\boldsymbol{\phi})^{-1} \delta\boldsymbol{\phi})}_{\dot{\boldsymbol{\phi}}'} \approx \underbrace{(\mathbf{J}(\boldsymbol{\phi}) + \delta\mathbf{J})^{-1}}_{\mathbf{J}(\boldsymbol{\phi}')} \underbrace{(\boldsymbol{\omega} + \delta\boldsymbol{\omega})}_{\boldsymbol{\omega}'}. \quad (7.233)$$

We obtain $\delta\mathbf{J}$ through a perturbation of $\mathbf{J}(\phi')$ directly:

$$\begin{aligned}\mathbf{J}(\phi') &= \int_0^1 \mathbf{C}'^\alpha d\alpha = \int_0^1 (\exp(\delta\phi^\wedge) \mathbf{C})^\alpha d\alpha \\ &\approx \int_0^1 (\mathbf{1} + (\mathbf{A}(\alpha, \phi) \delta\phi)^\wedge) \mathbf{C}^\alpha d\alpha \\ &= \underbrace{\int_0^1 \mathbf{C}^\alpha d\alpha}_{\mathbf{J}(\phi)} + \underbrace{\int_0^1 \alpha (\mathbf{J}(\alpha\phi) \mathbf{J}(\phi)^{-1} \delta\phi)^\wedge \mathbf{C}^\alpha d\alpha}_{\delta\mathbf{J}},\end{aligned}\quad (7.234)$$

where we have used the perturbed interpolation formula from Section 7.1.7. Manipulating the perturbed kinematics equation, we have

$$\begin{aligned}\dot{\phi} - \mathbf{J}(\phi)^{-1} \dot{\mathbf{J}}(\phi) \mathbf{J}(\phi)^{-1} \delta\phi + \mathbf{J}(\phi)^{-1} \delta\dot{\phi} \\ \approx (\mathbf{J}(\phi)^{-1} - \mathbf{J}(\phi)^{-1} \delta\mathbf{J} \mathbf{J}(\phi)^{-1}) (\boldsymbol{\omega} + \delta\boldsymbol{\omega}).\end{aligned}\quad (7.235)$$

Multiplying out, dropping the nominal solution, $\dot{\phi} = \mathbf{J}(\phi)^{-1} \boldsymbol{\omega}$, as well as products of small terms, we have

$$\delta\dot{\phi} = \dot{\mathbf{J}}(\phi) \mathbf{J}(\phi)^{-1} \delta\phi - \delta\mathbf{J} \dot{\phi} + \delta\boldsymbol{\omega}. \quad (7.236)$$

Substituting in the identity²⁴

$$\dot{\mathbf{J}}(\phi) - \boldsymbol{\omega}^\wedge \mathbf{J}(\phi) \equiv \frac{\partial \boldsymbol{\omega}}{\partial \phi}, \quad (7.237)$$

we have

$$\delta\dot{\phi} = \boldsymbol{\omega}^\wedge \delta\phi + \delta\boldsymbol{\omega} + \underbrace{\frac{\partial \boldsymbol{\omega}}{\partial \phi} \mathbf{J}(\phi)^{-1} \delta\phi - \delta\mathbf{J} \dot{\phi}}_{\text{extra term}}, \quad (7.238)$$

which is the same as the Lie group result for the perturbation kinematics, but with an extra term; it turns out this extra term is zero:

$$\begin{aligned}\frac{\partial \boldsymbol{\omega}}{\partial \phi} \mathbf{J}(\phi)^{-1} \delta\phi &= \left(\frac{\partial}{\partial \phi} (\mathbf{J}(\phi) \dot{\phi}) \right) \mathbf{J}(\phi)^{-1} \delta\phi \\ &= \left(\frac{\partial}{\partial \phi} \int_0^1 \mathbf{C}^\alpha \dot{\phi} d\alpha \right) \mathbf{J}(\phi)^{-1} \delta\phi = \int_0^1 \frac{\partial}{\partial \phi} (\mathbf{C}^\alpha \dot{\phi}) d\alpha \mathbf{J}(\phi)^{-1} \delta\phi \\ &= - \int_0^1 \alpha (\mathbf{C}^\alpha \dot{\phi})^\wedge \mathbf{J}(\alpha\phi) d\alpha \mathbf{J}(\phi)^{-1} \delta\phi \\ &= \underbrace{\int_0^1 \alpha (\mathbf{J}(\alpha\phi) \mathbf{J}(\phi)^{-1} \delta\phi)^\wedge \mathbf{C}^\alpha d\alpha}_{\delta\mathbf{J}} \dot{\phi} = \delta\mathbf{J} \dot{\phi},\end{aligned}\quad (7.239)$$

where we have used an identity derived back in Section 6.2.5 for the

²⁴ This identity is well known in the dynamics literature (Hughes, 1986).

derivative of a rotation matrix times a vector with respect to a three-parameter representation of rotation. Thus, our pair of equations is

$$\text{nominal kinematics: } \dot{\phi} = \mathbf{J}(\phi)^{-1}\omega, \quad (7.240\text{a})$$

$$\text{perturbation kinematics: } \delta\dot{\phi} = \omega^\wedge \delta\phi + \delta\omega, \quad (7.240\text{b})$$

which can be integrated separately and combined to provide the complete solution (approximately).

Solutions Commute

It is worth asking whether integrating the full solution is (approximately) equivalent to integrating the nominal and perturbation equations separately and then combining them. We show this for the Lie group kinematics. The perturbed solution will be given by:

$$\mathbf{C}'(t) = \mathbf{C}'(0) + \int_0^t \omega'(s)^\wedge \mathbf{C}'(s) ds. \quad (7.241)$$

Breaking this into nominal and perturbation parts, we have

$$\begin{aligned} \mathbf{C}'(t) &\approx (\mathbf{1} + \delta\phi(0)^\wedge) \mathbf{C}(0) + \int_0^t (\omega + \delta\omega)^\wedge (\mathbf{1} + \delta\phi(s)^\wedge) \mathbf{C}(s) ds \\ &\approx \underbrace{\mathbf{C}(0) + \int_0^t \omega(s)^\wedge \mathbf{C}(s) ds}_{\mathbf{C}(t)} \\ &\quad + \underbrace{\delta\phi(0)^\wedge \mathbf{C}(0) + \int_0^t (\omega(s)^\wedge \delta\phi(s)^\wedge \mathbf{C}(s) + \delta\omega(s)^\wedge \mathbf{C}(s)) ds}_{\delta\phi(t)^\wedge \mathbf{C}(t)} \\ &\approx (\mathbf{1} + \delta\phi(t)^\wedge) \mathbf{C}(t), \end{aligned} \quad (7.242)$$

which is the desired result. The rightmost integral on the second line can be computed by noting that

$$\begin{aligned} \frac{d}{dt} (\delta\phi^\wedge \mathbf{C}) &= \delta\dot{\phi}^\wedge \mathbf{C} + \delta\phi^\wedge \dot{\mathbf{C}} = (\underbrace{\omega^\wedge \delta\phi + \delta\omega}_{\text{perturbation}})^\wedge \mathbf{C} + \delta\phi^\wedge (\underbrace{\omega^\wedge \mathbf{C}}_{\text{nom.}}) \\ &= \omega^\wedge \delta\phi^\wedge \mathbf{C} - \delta\phi^\wedge \omega^\wedge \mathbf{C} + \delta\omega^\wedge \mathbf{C} + \delta\phi^\wedge \omega^\wedge \mathbf{C} \\ &= \omega^\wedge \delta\phi^\wedge \mathbf{C} + \delta\omega^\wedge \mathbf{C}, \end{aligned} \quad (7.243)$$

where we have used the nominal and perturbation kinematics.

Integrating the Solutions

In this section, we make some observations about integrating the nominal and perturbation kinematics. The nominal equation is nonlinear

and can be integrated numerically (using either the Lie group or Lie algebra equations). The perturbation kinematics,

$$\dot{\delta\phi}(t) = \omega(t)^\wedge \delta\phi(t) + \delta\omega(t), \quad (7.244)$$

is a LTV equation of the form

$$\dot{\mathbf{x}}(t) = \mathbf{A}(t) \mathbf{x}(t) + \mathbf{B}(t) \mathbf{u}(t). \quad (7.245)$$

The general solution to the initial value problem is given by

$$\mathbf{x}(t) = \Phi(t, 0) \mathbf{x}(0) + \int_0^t \Phi(t, s) \mathbf{B}(s) \mathbf{u}(s) ds, \quad (7.246)$$

where $\Phi(t, s)$ is called the *state transition matrix* and satisfies

$$\begin{aligned} \dot{\Phi}(t, s) &= \mathbf{A}(t) \Phi(t, s), \\ \Phi(t, t) &= \mathbf{1}. \end{aligned}$$

The state transition matrix always exists and is unique, but it cannot always be found analytically. Fortunately, for our particular perturbation equation, we can express the 3×3 state transition matrix analytically²⁵:

$$\Phi(t, s) = \mathbf{C}(t) \mathbf{C}(s)^T. \quad (7.247)$$

The solution is therefore given by

$$\delta\phi(t) = \mathbf{C}(t) \mathbf{C}(0)^T \delta\phi(0) + \mathbf{C}(t) \int_0^t \mathbf{C}(s)^T \delta\omega(s) ds. \quad (7.248)$$

We need the solution to the nominal equation, $\mathbf{C}(t)$, but this is readily available. To see this is indeed the correct solution, we can differentiate:

$$\begin{aligned} \dot{\delta\phi}(t) &= \dot{\mathbf{C}}(t) \mathbf{C}(0)^T \delta\phi(0) + \dot{\mathbf{C}}(t) \int_0^t \mathbf{C}(s)^T \delta\omega(s) ds + \mathbf{C}(t) \mathbf{C}(t)^T \delta\omega(t) \\ &= \omega(t)^\wedge \underbrace{\left(\mathbf{C}(t) \mathbf{C}(0)^T \delta\phi(0) + \mathbf{C}(t) \int_0^t \mathbf{C}(s)^T \delta\omega(s) ds \right)}_{\delta\phi(t)} + \delta\omega(t) \\ &= \omega(t)^\wedge \delta\phi(t) + \delta\omega(t), \end{aligned} \quad (7.249)$$

which is the original differential equation for $\delta\phi(t)$. We also see that our state transition matrix satisfies the required conditions:

$$\underbrace{\frac{d}{dt} (\mathbf{C}(t) \mathbf{C}(s)^T)}_{\dot{\Phi}(t, s)} = \dot{\mathbf{C}}(t) \mathbf{C}(s)^T = \omega(t)^\wedge \underbrace{\mathbf{C}(t) \mathbf{C}(s)^T}_{\Phi(t, s)}, \quad (7.250a)$$

$$\underbrace{\mathbf{C}(t) \mathbf{C}(t)^T}_{\Phi(t, t)} = \mathbf{1}. \quad (7.250b)$$

²⁵ The nominal rotation matrix, $\mathbf{C}(t)$, is the *fundamental matrix* of the state transition matrix.

Thus, we have everything we need to integrate the perturbation kinematics as long as we can also integrate the nominal kinematics.

7.2.4 Linearized Poses

We will only briefly summarize the perturbed kinematics for $SE(3)$ as they are quite similar to the $SO(3)$ case.

Lie Group

We will use the perturbation,

$$\mathbf{T}' = \exp(\delta\xi^\wedge) \mathbf{T} \approx (\mathbf{1} + \delta\xi^\wedge) \mathbf{T}, \quad (7.251)$$

with $\mathbf{T}', \mathbf{T} \in SE(3)$ and $\delta\xi \in \mathbb{R}^6$. The perturbed kinematics,

$$\dot{\mathbf{T}}' = \boldsymbol{\omega}'^\wedge \mathbf{T}', \quad (7.252)$$

can then be broken into nominal and perturbation kinematics:

$$\text{nominal kinematics: } \dot{\mathbf{T}} = \boldsymbol{\omega}^\wedge \mathbf{T}, \quad (7.253a)$$

$$\text{perturbation kinematics: } \delta\dot{\xi} = \boldsymbol{\omega}^\wedge \delta\xi + \delta\boldsymbol{\omega}, \quad (7.253b)$$

where $\boldsymbol{\omega}' = \boldsymbol{\omega} + \delta\boldsymbol{\omega}$. These can be integrated separately and combined to provide the complete solution (approximately).

Integrating the Solutions

The 6×6 transition matrix for the perturbation equation is

$$\Phi(t, s) = \mathcal{T}(t) \mathcal{T}(s)^{-1}, \quad (7.254)$$

where $\mathcal{T} = \text{Ad}(\mathbf{T})$. The solution for $\delta\xi(t)$ is

$$\delta\xi(t) = \mathcal{T}(t) \mathcal{T}(0)^{-1} \delta\xi(0) + \mathcal{T}(t) \int_0^t \mathcal{T}(s)^{-1} \delta\boldsymbol{\omega}(s) ds. \quad (7.255)$$

Differentiating recovers the perturbation kinematics, where we require the 6×6 version of the nominal kinematics in the derivation:

$$\dot{\mathcal{T}}(t) = \boldsymbol{\omega}(t)^\wedge \mathcal{T}(t), \quad (7.256)$$

which is equivalent to the 4×4 version.

With Dynamics

We can also perturb the joint kinematics/dynamics equations in (7.227). We consider perturbing all of the quantities around some operating points as follows:

$$\mathbf{T}' = \exp(\delta\xi^\wedge) \mathbf{T}, \quad \boldsymbol{\omega}' = \boldsymbol{\omega} + \delta\boldsymbol{\omega}, \quad \mathbf{a}' = \mathbf{a} + \delta\mathbf{a}, \quad (7.257)$$

so that the kinematics/dynamics are

$$\dot{\mathbf{T}}' = \boldsymbol{\varpi}'^\wedge \mathbf{T}', \quad (7.258a)$$

$$\dot{\boldsymbol{\varpi}}' = -\mathcal{M}^{-1} \boldsymbol{\varpi}'^{\wedge^T} \mathcal{M} \boldsymbol{\varpi}' + \mathbf{a}'. \quad (7.258b)$$

If we think of $\delta\mathbf{a}$ as an unknown noise input, then we would like to know how this turns into uncertainty on the pose and velocity variables through the chain of dynamics and kinematics. Substituting the perturbations into the motion models, we can separate into a (nonlinear) nominal motion model,

$$\text{nominal kinematics: } \dot{\mathbf{T}} = \boldsymbol{\varpi}^\wedge \mathbf{T}, \quad (7.259a)$$

$$\text{nominal dynamics: } \dot{\boldsymbol{\varpi}} = -\mathcal{M}^{-1} \boldsymbol{\varpi}^{\wedge^T} \mathcal{M} \boldsymbol{\varpi} + \mathbf{a}, \quad (7.259b)$$

and (linear) perturbation motion model,

$$\text{perturbation kinematics: } \dot{\delta\boldsymbol{\xi}} = \boldsymbol{\varpi}^\wedge \delta\boldsymbol{\xi} + \delta\boldsymbol{\varpi}, \quad (7.260a)$$

$$\text{perturbation dynamics: } \dot{\delta\boldsymbol{\varpi}} = \mathcal{M}^{-1} \left((\mathcal{M}\boldsymbol{\varpi})^{\wedge^T} - \boldsymbol{\varpi}^\wedge \mathcal{M} \right) \delta\boldsymbol{\varpi} + \delta\mathbf{a}, \quad (7.260b)$$

which we can write in combined matrix form:

$$\begin{bmatrix} \dot{\delta\boldsymbol{\xi}} \\ \dot{\delta\boldsymbol{\varpi}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\varpi}^\wedge & \mathbf{1} \\ \mathbf{0} & \mathcal{M}^{-1} \left((\mathcal{M}\boldsymbol{\varpi})^{\wedge^T} - \boldsymbol{\varpi}^\wedge \mathcal{M} \right) \end{bmatrix} \begin{bmatrix} \delta\boldsymbol{\xi} \\ \delta\boldsymbol{\varpi} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \delta\mathbf{a} \end{bmatrix}. \quad (7.261)$$

Finding the transition matrix for this LTV SDE may be difficult, but it can be integrated numerically.

7.3 Probability and Statistics

We have seen throughout this chapter that elements of matrix Lie groups do not satisfy some basic operations that we normally take for granted. This theme continues when working with random variables. For example, we often work with Gaussian random variables, which typically take the form,

$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma), \quad (7.262)$$

where $\mathbf{x} \in \mathbb{R}^N$ (i.e., \mathbf{x} lives in a vectorspace). An equivalent way to look at this is that \mathbf{x} comprises a ‘large’, noise-free component, $\boldsymbol{\mu}$, and a ‘small’, noisy component, $\boldsymbol{\epsilon}$, that is zero-mean:

$$\mathbf{x} = \boldsymbol{\mu} + \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \Sigma). \quad (7.263)$$

This arrangement works because all the quantities involved are vectors and the vectorspace is closed under the $+$ operation. Unfortunately, our matrix Lie groups are not closed under this type of addition, and so we need to think of a different way of defining random variables.

This section will introduce our definitions of random variables and

probability density functions (PDFs) for rotations and poses, and then present four examples of using our new probability and statistics. We follow the approach outlined by Barfoot and Furgale (2014), which is a practical method when uncertainty on rotation does not become too large. This approach was inspired by and builds on the works of Su and Lee (1991, 1992); Chirikjian and Kyatkin (2001); Smith et al. (2003); Wang and Chirikjian (2006, 2008); Chirikjian (2009); Wolfe et al. (2011); Long et al. (2012); Chirikjian and Kyatkin (2016). It must be noted that Chirikjian and Kyatkin (2001) (and the revision in (Chirikjian and Kyatkin, 2016)) explain how to represent and propagate PDFs on groups even when the uncertainty becomes large, whereas the discussion here is relevant only when uncertainty is reasonably small.

7.3.1 Gaussian Random Variables and PDFs

We will discuss general random variables and PDFs briefly, and then focus on Gaussians. We frame the main discussion in terms of rotations and then state the results for poses afterward.

Rotations

We have seen several times the dual nature of rotations/poses in the sense that they can be described in terms of a Lie group or a Lie algebra, each having advantages and disadvantages. Lie groups are nice because they are free of singularities but have constraints; this is also the form that is usually required in order to rotate/transform something in the real world. Lie algebras are nice because we can treat them as vectorspaces (for which there are many useful mathematical tools²⁶), and they are free of constraints, but we need to worry about singularities.

It seems logical to exploit the vectorspace character of a Lie algebra in defining our random variables for rotations and poses. In this way, we can leverage all the usual tools from probability and statistics, rather than starting over. Given this decision, and using (7.263) for inspiration, there are three possible ways to define a random variable for $SO(3)$ based on the different perturbation options:

	$SO(3)$	$\mathfrak{so}(3)$
left	$\mathbf{C} = \exp(\boldsymbol{\epsilon}_\ell^\wedge) \bar{\mathbf{C}}$	$\boldsymbol{\phi} \approx \boldsymbol{\mu} + \mathbf{J}_\ell^{-1}(\boldsymbol{\mu}) \boldsymbol{\epsilon}_\ell$
middle	$\mathbf{C} = \exp((\boldsymbol{\mu} + \boldsymbol{\epsilon}_m)^\wedge)$	$\boldsymbol{\phi} = \boldsymbol{\mu} + \boldsymbol{\epsilon}_m$
right	$\mathbf{C} = \bar{\mathbf{C}} \exp(\boldsymbol{\epsilon}_r^\wedge)$	$\boldsymbol{\phi} \approx \boldsymbol{\mu} + \mathbf{J}_r^{-1}(\boldsymbol{\mu}) \boldsymbol{\epsilon}_r$

where $\boldsymbol{\epsilon}_\ell, \boldsymbol{\epsilon}_m, \boldsymbol{\epsilon}_r \in \mathbb{R}^3$ are random variables in the usual (vectorspace) sense, $\boldsymbol{\mu} \in \mathbb{R}^3$ is a constant, and $\mathbf{C} = \exp(\boldsymbol{\phi}^\wedge), \bar{\mathbf{C}} = \exp(\boldsymbol{\mu}^\wedge) \in SO(3)$.

²⁶ Including probability and statistics.

In each of these three cases, we know through the surjective-only property of the exponential map and the closure property of Lie groups that we will ensure that $\mathbf{C} = \exp(\boldsymbol{\phi}^\wedge) \in SO(3)$. This idea of mapping a random variable onto a Lie group through the exponential map is sometimes informally referred to as ‘injecting’ noise onto the group, but this is misleading²⁷.

Looking to the Lie algebra versions of the perturbations, we can see the usual relationships between the left, middle, right: $\boldsymbol{\epsilon}_m \approx \mathbf{J}_\ell^{-1}(\boldsymbol{\mu}) \boldsymbol{\epsilon}_\ell \approx \mathbf{J}_r^{-1}(\boldsymbol{\mu}) \boldsymbol{\epsilon}_r$. Based on this, we might conclude that all the options are equally good. However, in the middle option, we must keep the nominal component of the variable in the Lie algebra as well as the perturbation, which means we will have to contend with the associated singularities. On the other hand, both the left and right perturbation approaches allow us to keep the nominal component of the variable in the Lie group. By convention, we will choose the left perturbation approach, but one could just as easily pick the right.

This approach to defining random variables for rotations/poses in some sense gets the best of both worlds. We can avoid singularities for the large, nominal part by keeping it in the Lie group, but we can exploit the constraint-free, vectorspace character of the Lie algebra for the small, noisy part. Since the noisy part is assumed to be small, it will tend to stay away from the singularities associated with the rotation-vector parameterization²⁸.

Thus, for $SO(3)$, a random variable, \mathbf{C} , will be of the form²⁹

$$\mathbf{C} = \exp(\boldsymbol{\epsilon}^\wedge) \bar{\mathbf{C}}, \quad (7.264)$$

where $\bar{\mathbf{C}} \in SO(3)$ is a ‘large’, noise-free, nominal rotation and $\boldsymbol{\epsilon} \in \mathbb{R}^3$ is a ‘small’, noisy component (i.e., it is just a regular, random variable from a vectorspace). This means that we can simply define a PDF for $\boldsymbol{\epsilon}$, and this will induce a PDF on $SO(3)$:

$$p(\boldsymbol{\epsilon}) \rightarrow p(\mathbf{C}). \quad (7.265)$$

We will mainly be concerned with Gaussian PDFs in our estimation

²⁷ Mathematically, *injection* means that at most one element of the Lie algebra should map to each element of the Lie group. As we have seen, the exponential map linking the Lie algebra to the Lie group is *surjective-only*, which means every element of the Lie algebra maps to some element of the Lie group and every element of the Lie group is mapped to from many elements of the Lie algebra. If we limit the rotation angle magnitude, $|\boldsymbol{\phi}| < \pi$, then the exponential map is *bijective*, meaning both surjective and injective (i.e., one-to-one and onto). However, we may not want to impose this limit, whereupon the injective property does not hold.

²⁸ This approach works reasonably well as long as the perturbation is small. If this is not the case, a more global approach to defining a random variable on Lie groups is required, but these are less well explored (Chirikjian and Kyatkin, 2001; Lee et al., 2008; Chirikjian and Kyatkin, 2016).

²⁹ We will drop the ℓ subscript from here to keep things clean.

problems, and in this case we let

$$p(\boldsymbol{\epsilon}) = \frac{1}{\sqrt{(2\pi)^3 \det(\boldsymbol{\Sigma})}} \exp\left(-\frac{1}{2} \boldsymbol{\epsilon}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\epsilon}\right), \quad (7.266)$$

or $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$. Note that we can now think of $\bar{\mathbf{C}}$ as the ‘mean’ rotation and $\boldsymbol{\Sigma}$ as the associated covariance.

By definition, $p(\boldsymbol{\epsilon})$ is a valid PDF, and so

$$\int p(\boldsymbol{\epsilon}) d\boldsymbol{\epsilon} = 1. \quad (7.267)$$

We deliberately avoid making the integration limits explicit because we have defined $\boldsymbol{\epsilon}$ to be Gaussian, which means it has probability mass out to infinity in all directions. However, we assume that most of the probability mass is encompassed in $|\boldsymbol{\epsilon}| < \pi$ for this to make sense. Referring back to Section 7.1.6, we know that we can relate an infinitesimal volume element in the Lie algebra to an infinitesimal volume element in the Lie group according to

$$d\mathbf{C} = |\det(\mathbf{J}(\boldsymbol{\epsilon}))| d\boldsymbol{\epsilon}, \quad (7.268)$$

where we note that due to our choice of using the left perturbation, the Jacobian, \mathbf{J} , is evaluated at $\boldsymbol{\epsilon}$ (which will hopefully be small) rather than at $\boldsymbol{\phi}$ (which could be large); this will hopefully keep \mathbf{J} very close to $\mathbf{1}$. We can use this to now work out the PDF that is induced on \mathbf{C} . We have that

$$1 = \int p(\boldsymbol{\epsilon}) d\boldsymbol{\epsilon} \quad (7.269)$$

$$= \int \frac{1}{\sqrt{(2\pi)^3 \det(\boldsymbol{\Sigma})}} \exp\left(-\frac{1}{2} \boldsymbol{\epsilon}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\epsilon}\right) d\boldsymbol{\epsilon} \quad (7.270)$$

$$= \underbrace{\int \frac{1}{\sqrt{(2\pi)^3 \det(\boldsymbol{\Sigma})}} \exp\left(-\frac{1}{2} \ln(\mathbf{C}\bar{\mathbf{C}}^T)^{\vee T} \boldsymbol{\Sigma}^{-1} \ln(\mathbf{C}\bar{\mathbf{C}}^T)^{\vee}\right)}_{p(\mathbf{C})} \frac{1}{|\det(\mathbf{J})|} d\mathbf{C}, \quad (7.271)$$

where we indicate the induced $p(\mathbf{C})$. It is important to realize that $p(\mathbf{C})$ looks like this due to our choice to define $p(\boldsymbol{\epsilon})$ directly³⁰.

A common method of defining the *mean rotation*, $\mathbf{M} \in SO(3)$, is the unique solution of the equation

$$\int \ln(\mathbf{C}\mathbf{M}^T)^{\vee} p(\mathbf{C}) d\mathbf{C} = \mathbf{0}. \quad (7.272)$$

Switching variables from \mathbf{C} to $\boldsymbol{\epsilon}$, this is equivalent to

$$\int \ln(\exp(\boldsymbol{\epsilon}^\wedge)\bar{\mathbf{C}}\mathbf{M}^T)^{\vee} p(\boldsymbol{\epsilon}) d\boldsymbol{\epsilon} = \mathbf{0}. \quad (7.273)$$

³⁰ It is also possible to work in the other direction by first defining $p(\mathbf{C})$ (Chirikjian, 2009).

Taking $\mathbf{M} = \bar{\mathbf{C}}$, we see that

$$\begin{aligned} \int \ln (\exp (\epsilon^\wedge) \bar{\mathbf{C}} \mathbf{M}^T)^\vee p(\epsilon) d\epsilon &= \int \ln (\exp (\epsilon^\wedge) \bar{\mathbf{C}} \bar{\mathbf{C}}^T)^\vee p(\epsilon) d\epsilon \\ &= \int \epsilon p(\epsilon) d\epsilon = E[\epsilon] = \mathbf{0}, \end{aligned} \quad (7.274)$$

which validates our logic in referring to $\bar{\mathbf{C}}$ as the mean earlier.

The corresponding *covariance*, Σ , computed about \mathbf{M} , can be defined as

$$\begin{aligned} \Sigma &= \int \ln (\exp (\epsilon^\wedge) \bar{\mathbf{C}} \mathbf{M}^T)^\vee \ln (\exp (\epsilon^\wedge) \bar{\mathbf{C}} \mathbf{M}^T)^{\vee T} p(\epsilon) d\epsilon \\ &= \int \ln (\exp (\epsilon^\wedge) \bar{\mathbf{C}} \bar{\mathbf{C}}^T)^\vee \ln (\exp (\epsilon^\wedge) \bar{\mathbf{C}} \bar{\mathbf{C}}^T)^{\vee T} p(\epsilon) d\epsilon \\ &= \int \epsilon \epsilon^T p(\epsilon) d\epsilon = E[\epsilon \epsilon^T], \end{aligned} \quad (7.275)$$

which implies that choosing $\epsilon \sim \mathcal{N}(\mathbf{0}, \Sigma)$ is a reasonable thing to do and matches nicely with the noise ‘injection’ procedure. In fact, all higher-order statistics defined in an analogous way will produce the statistics associated with ϵ as well.

As another advantage to this approach to representing random variables for rotations, consider what happens to our rotation random variables under a pure (deterministic) rotation mapping. Let $\mathbf{R} \in SO(3)$ be a constant rotation matrix that we apply to \mathbf{C} to create a new random variable, $\mathbf{C}' = \mathbf{R} \mathbf{C}$. With no approximation we have

$$\mathbf{C}' = \mathbf{R} \mathbf{C} = \mathbf{R} \exp (\epsilon^\wedge) \bar{\mathbf{C}} = \exp ((\mathbf{R} \epsilon)^\wedge) \mathbf{R} \bar{\mathbf{C}} = \exp (\epsilon'^\wedge) \bar{\mathbf{C}}', \quad (7.276)$$

where

$$\bar{\mathbf{C}}' = \mathbf{R} \bar{\mathbf{C}}, \quad \epsilon' = \mathbf{R} \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{R} \Sigma \mathbf{R}^T). \quad (7.277)$$

This is very appealing, as it allows us to carry out this common operation exactly.

Poses

Similarly to the rotation case, we choose to define a Gaussian random variable for poses as

$$\mathbf{T} = \exp (\epsilon^\wedge) \bar{\mathbf{T}}, \quad (7.278)$$

where $\bar{\mathbf{T}} \in SE(3)$ is a ‘large’ mean transformation and $\epsilon \in \mathbb{R}^6 \sim \mathcal{N}(\mathbf{0}, \Sigma)$ is a ‘small’ Gaussian random variable (i.e., in a vectorspace).

The *mean transformation*, $\mathbf{M} \in SE(3)$, is the unique solution of the equation,

$$\int \ln (\exp (\epsilon^\wedge) \bar{\mathbf{T}} \mathbf{M}^{-1})^\vee p(\epsilon) d\epsilon = \mathbf{0}. \quad (7.279)$$

Taking $\mathbf{M} = \bar{\mathbf{T}}$, we see that

$$\begin{aligned} \int \ln (\exp(\epsilon^\wedge) \bar{\mathbf{T}} \mathbf{M}^{-1})^\vee p(\epsilon) d\epsilon &= \int \ln (\exp(\epsilon^\wedge) \bar{\mathbf{T}} \bar{\mathbf{T}}^{-1})^\vee p(\epsilon) d\epsilon \\ &= \int \epsilon p(\epsilon) d\epsilon = E[\epsilon] = \mathbf{0}, \end{aligned} \quad (7.280)$$

which validates our logic in referring to $\bar{\mathbf{T}}$ as the mean.

The corresponding covariance, Σ , computed about \mathbf{M} , can be defined as

$$\begin{aligned} \Sigma &= \int \ln (\exp(\epsilon^\wedge) \bar{\mathbf{T}} \mathbf{M}^{-1})^\vee \ln (\exp(\epsilon^\wedge) \bar{\mathbf{T}} \mathbf{M}^{-1})^{\vee^T} p(\epsilon) d\epsilon \\ &= \int \ln (\exp(\epsilon^\wedge) \bar{\mathbf{T}} \bar{\mathbf{T}}^{-1})^\vee \ln (\exp(\epsilon^\wedge) \bar{\mathbf{T}} \bar{\mathbf{T}}^{-1})^{\vee^T} p(\epsilon) d\epsilon \\ &= \int \epsilon \epsilon^T p(\epsilon) d\epsilon = E[\epsilon \epsilon^T], \end{aligned} \quad (7.281)$$

which implies that choosing $\epsilon \sim \mathcal{N}(\mathbf{0}, \Sigma)$ is a reasonable thing to do and matches nicely with the noise ‘injection’ procedure. In fact, all higher-order statistics defined in an analogous way will produce the statistics associated with ϵ as well.

Again, consider what happens to our transformation random variables under a pure (deterministic) transformation mapping. Let $\mathbf{R} \in SE(3)$ be a constant transformation matrix that we apply to \mathbf{T} to create a new random variable, $\mathbf{T}' = \mathbf{R} \mathbf{T}$. With no approximation we have

$$\mathbf{T}' = \mathbf{R} \mathbf{T} = \mathbf{R} \exp(\epsilon^\wedge) \bar{\mathbf{T}} = \exp((\mathbf{R}\epsilon)^\wedge) \mathbf{R} \bar{\mathbf{T}} = \exp(\epsilon'^\wedge) \bar{\mathbf{T}}', \quad (7.282)$$

where

$$\bar{\mathbf{T}}' = \mathbf{R} \bar{\mathbf{T}}, \quad \epsilon' = \mathbf{R} \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{R} \Sigma \mathbf{R}^T), \quad (7.283)$$

and $\mathbf{R} = \text{Ad}(\mathbf{R})$.

7.3.2 Uncertainty on a Rotated Vector

Consider the simple mapping from rotation to position given by

$$\mathbf{y} = \mathbf{C}\mathbf{x}, \quad (7.284)$$

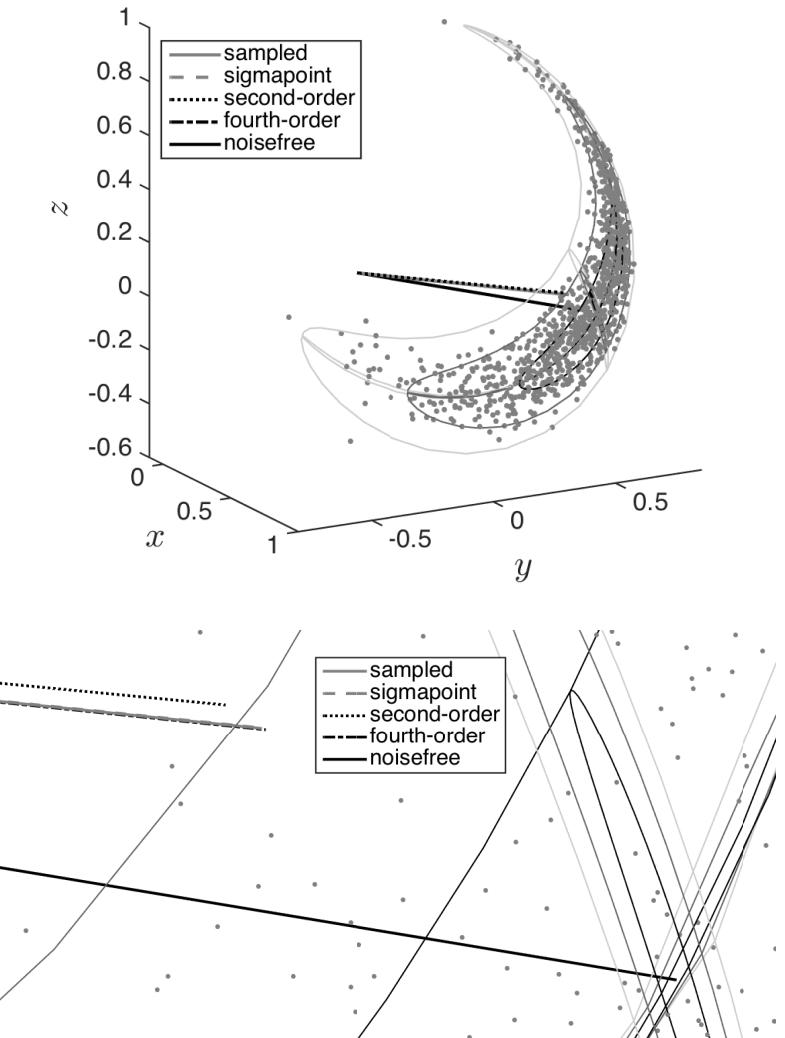
where $\mathbf{x} \in \mathbb{R}^3$ is a constant and

$$\mathbf{C} = \exp(\epsilon^\wedge) \bar{\mathbf{C}}, \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \Sigma). \quad (7.285)$$

Figure 7.4 shows what the resulting density over \mathbf{y} looks like for some particular values of $\bar{\mathbf{C}}$ and Σ . We see that, as expected, the samples live on a sphere whose radius is $|\mathbf{x}|$ since rotations preserve length.

We might be interested in computing $E[\mathbf{y}]$ in the vectorspace \mathbb{R}^3 (i.e.,

Figure 7.4
 Depiction of uncertainty on a vector $\mathbf{y} = \mathbf{Cx} \in \mathbb{R}^3$, where \mathbf{x} is constant and $\mathbf{C} = \exp(\epsilon^\wedge) \bar{\mathbf{C}}$, $\phi \sim \mathcal{N}(\mathbf{0}, \Sigma)$ is a random variable. The dots show samples of the resulting density over \mathbf{y} . The contours (of varying darkness) show the 1, 2, 3 standard deviation equiprobable contours of ϵ mapped to \mathbf{y} . The solid black line is the noisefree vector, $\bar{\mathbf{y}} = \bar{\mathbf{C}}\mathbf{x}$. The grey, dashed, dotted, and dash-dotted lines show various estimates of $E[\mathbf{y}]$ using brute-force sampling, the sigma point transformation, a second-order method, and a fourth-order method.



not exploiting special knowledge that \mathbf{y} must have length $|\mathbf{x}|$). We can imagine three ways of doing this:

- (i) Drawing a large number of random samples and then averaging.
- (ii) Using the sigma point transformation.
- (iii) An analytical approximation.

For (iii), we consider expanding \mathbf{C} in terms of ϵ so that

$$\mathbf{y} = \mathbf{Cx} = \left(\mathbf{1} + \epsilon^\wedge + \frac{1}{2}\epsilon^\wedge\epsilon^\wedge + \frac{1}{6}\epsilon^\wedge\epsilon^\wedge\epsilon^\wedge + \frac{1}{24}\epsilon^\wedge\epsilon^\wedge\epsilon^\wedge\epsilon^\wedge + \dots \right) \bar{\mathbf{C}}\mathbf{x}. \quad (7.286)$$

Since ϵ is Gaussian, the odd terms average to zero such that

$$E[\mathbf{y}] = \left(\mathbf{1} + \frac{1}{2} E[\epsilon^\wedge \epsilon^\wedge] + \frac{1}{24} E[\epsilon^\wedge \epsilon^\wedge \epsilon^\wedge \epsilon^\wedge] + \dots \right) \bar{\mathbf{C}}\mathbf{x}. \quad (7.287)$$

Going term by term, we have

$$\begin{aligned} E[\epsilon^\wedge \epsilon^\wedge] &= E[-(\epsilon^T \epsilon) \mathbf{1} + \epsilon \epsilon^T] = -\text{tr}(E[\epsilon \epsilon^T]) \mathbf{1} + E[\epsilon \epsilon^T] \\ &= -\text{tr}(\Sigma) \mathbf{1} + \Sigma, \end{aligned} \quad (7.288)$$

and

$$\begin{aligned} E[\epsilon^\wedge \epsilon^\wedge \epsilon^\wedge \epsilon^\wedge] &= E[-(\epsilon^T \epsilon) \epsilon^\wedge \epsilon^\wedge] \\ &= E[-(\epsilon^T \epsilon) (-(\epsilon^T \epsilon) \mathbf{1} + \epsilon \epsilon^T)] \\ &= \text{tr}(E[(\epsilon^T \epsilon) \epsilon \epsilon^T]) \mathbf{1} - E[(\epsilon^T \epsilon) \epsilon \epsilon^T] \\ &= \text{tr}(\Sigma (\text{tr}(\Sigma) \mathbf{1} + 2\Sigma)) \mathbf{1} - \Sigma (\text{tr}(\Sigma) \mathbf{1} + 2\Sigma) \\ &= ((\text{tr}(\Sigma))^2 + 2 \text{tr}(\Sigma^2)) \mathbf{1} - \Sigma (\text{tr}(\Sigma) \mathbf{1} + 2\Sigma), \end{aligned} \quad (7.289)$$

where we have used the multivariate version of Isserlis' theorem. Higher-order terms are also possible, but to fourth order in ϵ , we have

$$\begin{aligned} E[\mathbf{y}] &\approx \left(\mathbf{1} + \frac{1}{2} (-\text{tr}(\Sigma) \mathbf{1} + \Sigma) \right. \\ &\quad \left. + \frac{1}{24} \left(((\text{tr}(\Sigma))^2 + 2 \text{tr}(\Sigma^2)) \mathbf{1} - \Sigma (\text{tr}(\Sigma) \mathbf{1} + 2\Sigma) \right) \right) \bar{\mathbf{C}}\mathbf{x}. \end{aligned} \quad (7.290)$$

We refer to the method keeping terms to second order in ϵ as ‘second-order’ and the method keeping terms to fourth order in ϵ as ‘fourth-order’ in Figure 7.4. The fourth-order method is very comparable to the sigmapoint method and random sampling.

7.3.3 Compounding Poses

In this section, we investigate the problem of compounding two poses, each with associated uncertainty, as depicted in Figure 7.5.

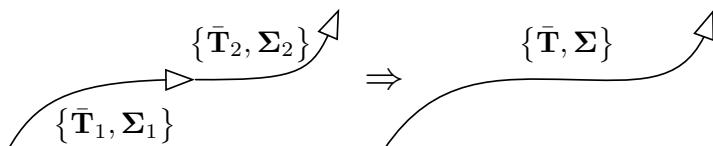


Figure 7.5
Combining a chain
of two poses into a
single compound
pose.

Theory

Consider two noisy poses, \mathbf{T}_1 and \mathbf{T}_2 ; we keep track of their nominal values and associated uncertainties:

$$\{\bar{\mathbf{T}}_1, \Sigma_1\}, \quad \{\bar{\mathbf{T}}_2, \Sigma_2\}. \quad (7.291)$$

Suppose now we let

$$\mathbf{T} = \mathbf{T}_1 \mathbf{T}_2, \quad (7.292)$$

as depicted in Figure 7.5. What is $\{\bar{\mathbf{T}}, \Sigma\}$? Under our perturbation scheme we have,

$$\exp(\epsilon^\wedge) \bar{\mathbf{T}} = \exp(\epsilon_1^\wedge) \bar{\mathbf{T}}_1 \exp(\epsilon_2^\wedge) \bar{\mathbf{T}}_2. \quad (7.293)$$

Moving all the uncertain factors to the left side, we have

$$\exp(\epsilon^\wedge) \bar{\mathbf{T}} = \exp(\epsilon_1^\wedge) \exp((\bar{\mathcal{T}}_1 \epsilon_2)^\wedge) \bar{\mathbf{T}}_1 \bar{\mathbf{T}}_2, \quad (7.294)$$

where $\bar{\mathcal{T}}_1 = \text{Ad}(\bar{\mathbf{T}}_1)$. If we let

$$\bar{\mathbf{T}} = \bar{\mathbf{T}}_1 \bar{\mathbf{T}}_2, \quad (7.295)$$

we are left with

$$\exp(\epsilon^\wedge) = \exp(\epsilon_1^\wedge) \exp((\bar{\mathcal{T}}_1 \epsilon_2)^\wedge). \quad (7.296)$$

Letting $\epsilon'_2 = \bar{\mathcal{T}}_1 \epsilon_2$, we can apply the BCH formula to find

$$\epsilon = \epsilon_1 + \epsilon'_2 + \frac{1}{2} \epsilon_1^\wedge \epsilon'_2 + \frac{1}{12} \epsilon_1^\wedge \epsilon_1^\wedge \epsilon'_2 + \frac{1}{12} \epsilon_2'^\wedge \epsilon_2'^\wedge \epsilon_1 - \frac{1}{24} \epsilon_2'^\wedge \epsilon_1^\wedge \epsilon_1^\wedge \epsilon'_2 + \dots \quad (7.297)$$

For our approach to hold, we require that $E[\epsilon] = \mathbf{0}$. Assuming that $\epsilon_1 \sim \mathcal{N}(\mathbf{0}, \Sigma_1)$ and $\epsilon'_2 \sim \mathcal{N}(\mathbf{0}, \Sigma'_2)$ are uncorrelated with one another, we have

$$E[\epsilon] = -\frac{1}{24} E[\epsilon_2'^\wedge \epsilon_1^\wedge \epsilon_1^\wedge \epsilon'_2] + O(\epsilon^6), \quad (7.298)$$

since everything except the fourth-order term has zero mean. Thus, to third order, we can safely assume that $E[\epsilon] = \mathbf{0}$, and thus (7.295) seems to be a reasonable way to compound the mean transformations³¹.

The next task is to compute $\Sigma = E[\epsilon \epsilon^T]$. Multiplying out to fourth

³¹ It is also possible to show that the fourth-order term has zero mean, $E[\epsilon_2'^\wedge \epsilon_1^\wedge \epsilon_1^\wedge \epsilon'_2] = \mathbf{0}$, if Σ_1 is of the special form

$$\Sigma_1 = \begin{bmatrix} \Sigma_{1,\rho\rho} & \mathbf{0} \\ \mathbf{0} & \sigma_{1,\phi\phi}^2 \mathbf{1} \end{bmatrix},$$

where the ρ and ϕ subscripts indicate a partitioning of the covariance into the translation and rotation components, respectively. This is a common situation for Σ_1 when we are, for example, propagating uncertainty on velocity through the kinematics equations presented for $SE(3)$; from (7.222) we have, $\mathbf{T}_1 = \exp((t_2 - t_1)\boldsymbol{\varpi}^\wedge)$, where $\boldsymbol{\varpi}$ is the (noisy) generalized velocity. In this case, we are justified in assuming $E[\epsilon] = \mathbf{0}$ all the way out to fifth order (and possibly further).

order, we have

$$\begin{aligned} E[\epsilon\epsilon^T] &\approx E\left[\epsilon_1\epsilon_1^T + \epsilon'_2\epsilon'^T_2 + \frac{1}{4}\epsilon_1^\lambda(\epsilon'_2\epsilon'^T_2)\epsilon_1^{\lambda T}\right. \\ &\quad + \frac{1}{12}\left((\epsilon_1^\lambda\epsilon_1^\lambda)(\epsilon'_2\epsilon'^T_2) + (\epsilon'_2\epsilon'^T_2)(\epsilon_1^\lambda\epsilon_1^\lambda)^T\right. \\ &\quad \left.\left. + (\epsilon_2'^\lambda\epsilon_2'^\lambda)(\epsilon_1\epsilon_1^T) + (\epsilon_2'^\lambda\epsilon_2'^\lambda)(\epsilon_1\epsilon_1^T)\right)\right] \quad (7.299) \end{aligned}$$

where we have omitted showing any terms that have an odd power in either ϵ_1 or ϵ'_2 since these will by definition have expectation zero. This expression may look daunting, but we can take it term by term. To save space, we define and make use of the following two linear operators:

$$\langle\langle \mathbf{A} \rangle\rangle = -\text{tr}(\mathbf{A}) \mathbf{1} + \mathbf{A}, \quad (7.300a)$$

$$\langle\langle \mathbf{A}, \mathbf{B} \rangle\rangle = \langle\langle \mathbf{A} \rangle\rangle \langle\langle \mathbf{B} \rangle\rangle + \langle\langle \mathbf{B} \mathbf{A} \rangle\rangle, \quad (7.300b)$$

with $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n \times n}$. These provide the useful identity,

$$-\mathbf{u}^\wedge \mathbf{A} \mathbf{v}^\wedge \equiv \langle\langle \mathbf{v} \mathbf{u}^T, \mathbf{A}^T \rangle\rangle, \quad (7.301)$$

where $\mathbf{u}, \mathbf{v} \in \mathbb{R}^3$ and $\mathbf{A} \in \mathbb{R}^{3 \times 3}$. Making use of this repeatedly, we have out to fourth order,

$$E[\epsilon_1\epsilon_1^T] = \Sigma_1 = \begin{bmatrix} \Sigma_{1,\rho\rho} & \Sigma_{1,\rho\phi} \\ \Sigma_{1,\rho\phi}^T & \Sigma_{1,\phi\phi} \end{bmatrix}, \quad (7.302a)$$

$$E[\epsilon'_2\epsilon'^T_2] = \Sigma'_2 = \begin{bmatrix} \Sigma'_{2,\bar{\rho}\bar{\rho}} & \Sigma'_{2,\bar{\rho}\bar{\phi}} \\ \Sigma'_{2,\bar{\rho}\bar{\phi}}^T & \Sigma'_{2,\bar{\phi}\bar{\phi}} \end{bmatrix} = \bar{\mathcal{T}}_1 \Sigma_2 \bar{\mathcal{T}}_1^T, \quad (7.302b)$$

$$E[\epsilon_1^\lambda\epsilon_1^\lambda] = \mathcal{A}_1 = \begin{bmatrix} \langle\langle \Sigma_{1,\phi\phi} \rangle\rangle & \langle\langle \Sigma_{1,\rho\phi} + \Sigma_{1,\rho\phi}^T \rangle\rangle \\ \mathbf{0} & \langle\langle \Sigma_{1,\phi\phi} \rangle\rangle \end{bmatrix}, \quad (7.302c)$$

$$E[\epsilon_2'^\lambda\epsilon_2'^\lambda] = \mathcal{A}'_2 = \begin{bmatrix} \langle\langle \Sigma'_{2,\phi\phi} \rangle\rangle & \langle\langle \Sigma'_{2,\rho\phi} + \Sigma'_{2,\rho\phi}^T \rangle\rangle \\ \mathbf{0} & \langle\langle \Sigma'_{2,\phi\phi} \rangle\rangle \end{bmatrix}, \quad (7.302d)$$

$$E[\epsilon_1^\lambda(\epsilon'_2\epsilon'^T_2)\epsilon_1^{\lambda T}] = \mathcal{B} = \begin{bmatrix} \mathbf{B}_{\rho\rho} & \mathbf{B}_{\rho\phi} \\ \mathbf{B}_{\rho\phi}^T & \mathbf{B}_{\phi\phi} \end{bmatrix}, \quad (7.302e)$$

where

$$\begin{aligned} \mathbf{B}_{\rho\rho} &= \langle\langle \Sigma_{1,\phi\phi}, \Sigma'_{2,\rho\rho} \rangle\rangle + \langle\langle \Sigma_{1,\rho\phi}^T, \Sigma'_{2,\rho\phi} \rangle\rangle \\ &\quad + \langle\langle \Sigma_{1,\rho\phi}, \Sigma'_{2,\rho\phi}^T \rangle\rangle + \langle\langle \Sigma_{1,\rho\rho}, \Sigma'_{2,\phi\phi} \rangle\rangle, \quad (7.303a) \end{aligned}$$

$$\mathbf{B}_{\rho\phi} = \langle\langle \Sigma_{1,\phi\phi}, \Sigma'_{2,\rho\phi} \rangle\rangle + \langle\langle \Sigma_{1,\rho\phi}^T, \Sigma'_{2,\phi\phi} \rangle\rangle, \quad (7.303b)$$

$$\mathbf{B}_{\phi\phi} = \langle\langle \Sigma_{1,\phi\phi}, \Sigma'_{2,\phi\phi} \rangle\rangle. \quad (7.303c)$$

The resulting covariance is then

$$\boldsymbol{\Sigma}_{\text{4th}} \approx \underbrace{\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}'_2}_{\boldsymbol{\Sigma}_{\text{2nd}}} + \underbrace{\frac{1}{4} \mathbf{B} + \frac{1}{12} (\mathbf{A}_1 \boldsymbol{\Sigma}'_2 + \boldsymbol{\Sigma}'_2 \mathbf{A}_1^T + \mathbf{A}'_2 \boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_1 \mathbf{A}'_2^T)}_{\text{additional fourth-order terms}}, \quad (7.304)$$

correct to fourth order³². This result is essentially the same as that of Wang and Chirikjian (2008) but worked out for our slightly different PDF; it is important to note that while our method is fourth order in the perturbation variables, it is only second order in the covariance (same as Wang and Chirikjian (2008)). Chirikjian and Kyatkin (2016) provide helpful insight on the relationship between these results. In summary, to compound two poses, we propagate the mean using (7.295) and the covariance using (7.304).

Sigmapoint Method

We can also make use of the sigmapoint transformation (Julier and Uhlmann, 1996) to pass uncertainty through the compound pose change. In this section, we tailor this to our specific type of $SE(3)$ perturbation. Our approach to handling sigmapoints is quite similar to that taken by Hertzberg et al. (2013) and also Brookshire and Teller (2012). In our case, we begin by approximating the joint input Gaussian using a finite number of samples, $\{\mathbf{T}_{1,\ell}, \mathbf{T}_{2,\ell}\}$:

$$\begin{aligned} \mathbf{L} \mathbf{L}^T &= \text{diag}(\boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2), \quad (\text{Cholesky decomposition; } \mathbf{L} \text{ lower-triangular}) \\ \boldsymbol{\psi}_\ell &= \sqrt{\lambda} \text{col}_\ell \mathbf{L}, \quad \ell = 1 \dots L, \\ \boldsymbol{\psi}_{\ell+L} &= -\sqrt{\lambda} \text{col}_\ell \mathbf{L}, \quad \ell = 1 \dots L, \\ \begin{bmatrix} \boldsymbol{\epsilon}_{1,\ell} \\ \boldsymbol{\epsilon}_{2,\ell} \end{bmatrix} &= \boldsymbol{\psi}_\ell, \quad \ell = 1 \dots 2L, \\ \mathbf{T}_{1,\ell} &= \exp(\boldsymbol{\epsilon}_{1,\ell}^\wedge) \bar{\mathbf{T}}_1, \quad \ell = 1 \dots 2L, \\ \mathbf{T}_{2,\ell} &= \exp(\boldsymbol{\epsilon}_{2,\ell}^\wedge) \bar{\mathbf{T}}_2, \quad \ell = 1 \dots 2L, \end{aligned}$$

where λ is a user-definable scaling constant³³ and $L = 12$. We then pass each of these samples through the compound pose change and compute the difference from the mean:

$$\boldsymbol{\epsilon}_\ell = \ln(\mathbf{T}_{1,\ell} \mathbf{T}_{2,\ell} \bar{\mathbf{T}}^{-1}), \quad \ell = 1 \dots 2L. \quad (7.305)$$

These are combined to create the output covariance according to

$$\boldsymbol{\Sigma}_{\text{sp}} = \frac{1}{2\lambda} \sum_{\ell=1}^{2L} \boldsymbol{\epsilon}_\ell \boldsymbol{\epsilon}_\ell^T. \quad (7.306)$$

³² The sixth order terms require a lot more work, but it is possible to compute them using Isserlis' theorem.

³³ For all experiments in this section, we used $\lambda = 1$; we need to ensure that the sigmapoints associated with the rotational degrees of freedom have length less than π to avoid numerical problems.

Note, we have assumed that the output sigmapoint samples have zero mean in this formula, to be consistent with our mean propagation. Interestingly, this turns out to be algebraically equivalent to the second-order method (from the previous section) for this particular nonlinearity since the noise sources on \mathbf{T}_1 and \mathbf{T}_2 are assumed to be uncorrelated.

Simple Compound Example

In this section, we present a simple qualitative example of pose compounding and in Section 7.3.3 we carry out a more quantitative study on a different setup. To see the qualitative difference between the second- and fourth-order methods, let us consider the case of compounding transformations many times in a row:

$$\exp(\boldsymbol{\epsilon}_K^\wedge) \bar{\mathbf{T}}_K = \left(\prod_{k=1}^K \exp(\boldsymbol{\epsilon}^\wedge) \bar{\mathbf{T}} \right) \exp(\boldsymbol{\epsilon}_0^\wedge) \bar{\mathbf{T}}_0. \quad (7.307)$$

As discussed earlier, this can be viewed as a discrete-time integration of the $SE(3)$ kinematic equations as in (7.222). To keep things simple, we make the following assumptions:

$$\bar{\mathbf{T}}_0 = \mathbf{1}, \quad \boldsymbol{\epsilon}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{0}), \quad (7.308a)$$

$$\bar{\mathbf{T}} = \begin{bmatrix} \bar{\mathbf{C}} & \bar{\mathbf{r}} \\ \mathbf{0}^T & 1 \end{bmatrix}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}), \quad (7.308b)$$

$$\bar{\mathbf{C}} = \mathbf{1}, \quad \bar{\mathbf{r}} = \begin{bmatrix} r \\ 0 \\ 0 \end{bmatrix}, \quad \boldsymbol{\Sigma} = \text{diag}(0, 0, 0, 0, 0, \sigma^2). \quad (7.308c)$$

Although this example uses our three-dimensional tools, it is confined to a plane for the purpose of illustration and ease of plotting; it corresponds to a rigid body moving along the x -axis but with some uncertainty only on the rotational velocity about the z -axis. This could model a unicycle robot driving in the plane with constant translational speed and slightly uncertain rotational speed (centered about zero). We are interested in how the covariance matrix fills in over time.

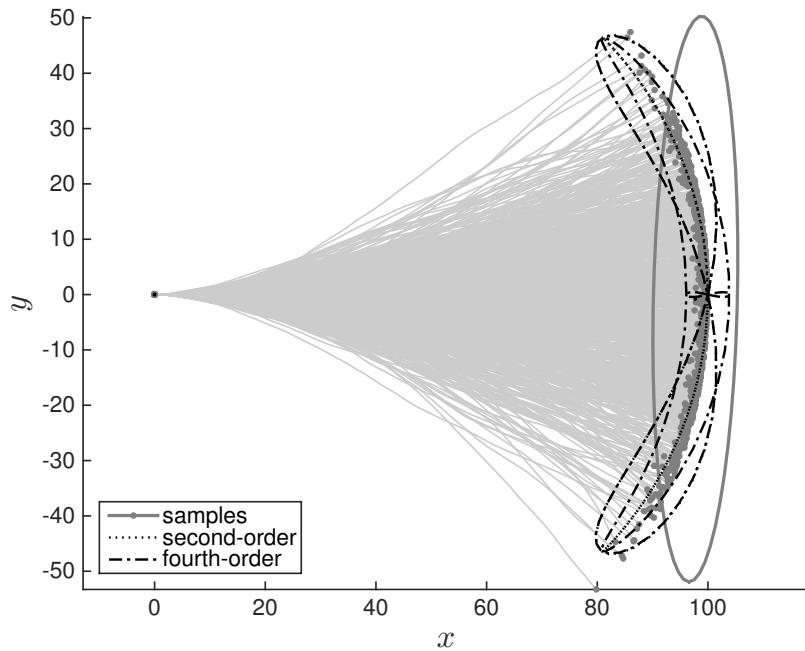
According to the second-order scheme, we have

$$\bar{\mathbf{T}}_K = \begin{bmatrix} 1 & 0 & 0 & Kr \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (7.309a)$$

$$\boldsymbol{\Sigma}_K = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{K(K-1)(2K-1)}{6}r^2\sigma^2 & 0 & 0 & -\frac{K(K-1)}{2}r\sigma^2 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{K(K-1)}{2}r\sigma^2 & 0 & 0 & K\sigma^2 \end{bmatrix}, \quad (7.309b)$$

Figure 7.6

Example of compounding $K = 100$ uncertain transformations (Section 7.3.3). The light grey lines and grey dots show 1000 individual sampled trajectories starting from $(0, 0)$ and moving nominally to the right at constant translational speed, but with some uncertainty on rotational velocity. The gray 1-sigma covariance ellipse is simply fitted to the samples to show what keeping xy -covariance relative to the start looks like. The black dotted (second-order) and dash-dotted (fourth-order) lines are the principal great circles of the 1-sigma covariance ellipsoid, given by Σ_K , mapped to the xy -plane. Looking to the area $(95, 0)$, corresponding to straight ahead, the fourth-order scheme has some non-zero uncertainty (as do the samples), whereas the second-order scheme does not. We used $r = 1$ and $\sigma = 0.03$.



where we see that the top-left entry of Σ_K , corresponding to uncertainty in the x -direction, does not have any growth of uncertainty. However, in the fourth-order scheme, the fill-in pattern is such that the top-left entry is non-zero. This happens for several reasons, but mainly through the $\mathbf{B}_{\rho\rho}$ submatrix of \mathbf{B} . This leaking of uncertainty into an additional degree of freedom cannot be captured by keeping only the second-order terms. Figure 7.6 provides a numerical example of this effect. It shows that both the second- and fourth-order schemes do a good job of representing the ‘banana’-like density over poses, as discussed by Long et al. (2012). However, the fourth-order scheme has some finite uncertainty in the straight-ahead direction (as do the sampled trajectories), while the second-order scheme does not.

Compound Experiment

To quantitatively evaluate the pose-compounding techniques, we ran a second numerical experiment in which we compounded two poses,

including their associated covariance matrices:

$$\bar{\mathbf{T}}_1 = \exp(\bar{\boldsymbol{\xi}}_1^\wedge), \quad \bar{\boldsymbol{\xi}}_1 = [0 \ 2 \ 0 \ \pi/6 \ 0 \ 0]^T, \\ \boldsymbol{\Sigma}_1 = \alpha \times \text{diag} \left\{ 10, 5, 5, \frac{1}{2}, 1, \frac{1}{2} \right\}, \quad (7.310a)$$

$$\bar{\mathbf{T}}_2 = \exp(\bar{\boldsymbol{\xi}}_2^\wedge), \quad \bar{\boldsymbol{\xi}}_2 = [0 \ 0 \ 1 \ 0 \ \pi/4 \ 0]^T, \\ \boldsymbol{\Sigma}_2 = \alpha \times \text{diag} \left\{ 5, 10, 5, \frac{1}{2}, \frac{1}{2}, 1 \right\}, \quad (7.310b)$$

where $\alpha \in [0, 1]$ is a scaling parameter that increases the magnitude of the input covariances parametrically.

We compounded these two poses according to (7.293), which results in a mean of $\bar{\mathbf{T}} = \bar{\mathbf{T}}_1 \bar{\mathbf{T}}_2$. The covariance, $\boldsymbol{\Sigma}$, was computed using four methods:

- (i) *Monte Carlo*: We drew a large number, $M = 1,000,000$, of random samples ($\boldsymbol{\epsilon}_{m_1}$ and $\boldsymbol{\epsilon}_{m_2}$) from the input covariance matrices, compounded the resulting transformations, and computed the covariance as $\boldsymbol{\Sigma}_{mc} = \frac{1}{M} \sum_{m=1}^M \boldsymbol{\epsilon}_m \boldsymbol{\epsilon}_m^T$ with $\boldsymbol{\epsilon}_m = \ln(\mathbf{T}_m \bar{\mathbf{T}}^{-1})^\vee$ and $\mathbf{T}_m = \exp(\boldsymbol{\epsilon}_{m_1}^\wedge) \bar{\mathbf{T}}_1 \exp(\boldsymbol{\epsilon}_{m_2}^\wedge) \bar{\mathbf{T}}_2$. This slow-but-accurate approach served as our benchmark to which the other three much faster methods were compared.
- (ii) *Second-Order*: We used the second-order method described above to compute $\boldsymbol{\Sigma}_{2nd}$.
- (iii) *Fourth-Order*: We used the fourth-order method described above to compute $\boldsymbol{\Sigma}_{4th}$.
- (iv) *Sigmapoint*: We used the sigmapoint transformation described above to compute $\boldsymbol{\Sigma}_{sp}$.

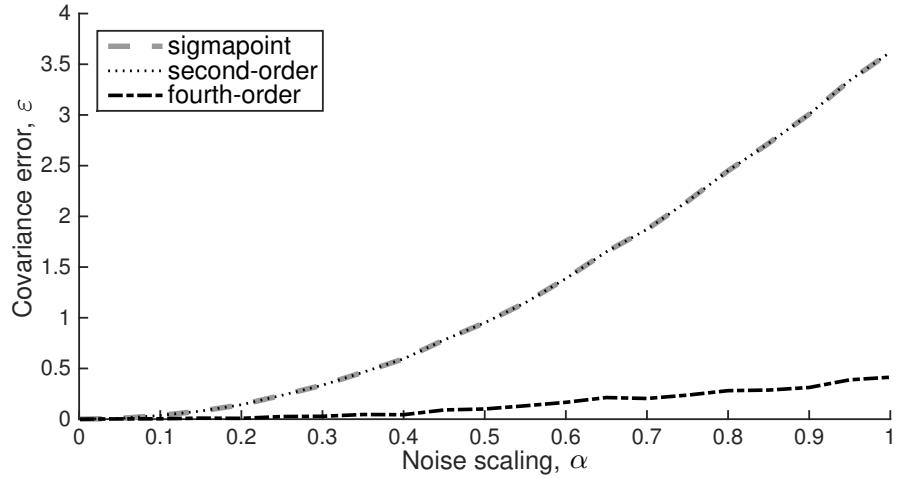
We compared each of the last three covariance matrices to the Monte Carlo one using the *Frobenius norm*:

$$\varepsilon = \sqrt{\text{tr} \left((\boldsymbol{\Sigma} - \boldsymbol{\Sigma}_{mc})^T (\boldsymbol{\Sigma} - \boldsymbol{\Sigma}_{mc}) \right)}.$$

Figure 7.7 shows that for small input covariance matrices (i.e., α small) there is very little difference between the various methods and the errors are all low compared to our benchmark. However, as we increase the magnitude of the input covariances, all the methods get worse, with the fourth-order method faring the best by about a factor of seven based on our error metric. Note that since α is scaling the covariance, the applied noise is increasing quadratically.

The second-order method and the sigmapoint method have indistinguishable performance, as they are algebraically equivalent. The fourth-order method goes beyond both of these by considering higher-order

Figure 7.7
 Results from Compound Experiment: error, ε , in computing covariance associated with compounding two poses using three methods, as compared to Monte Carlo. The sigmapoint and second-order methods are algebraically equivalent for this problem and thus appear the same on the plot. The input covariances were gradually scaled up via the parameter, α , highlighting the improved performance of the fourth-order method.



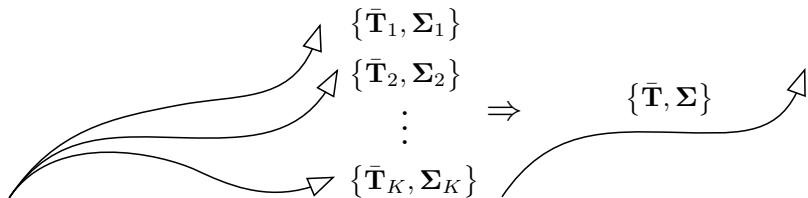
terms in the input covariance matrices. We did not compare the computational costs of the various methods as they are all extremely efficient as compared to Monte Carlo.

It is also worth noting that our ability to correctly keep track of uncertainties on $SE(3)$ decreases with increasing uncertainty. This can be seen directly in Figure 7.7, as error increases with increasing uncertainty. This suggests that it may be wise to use only relative pose variables to keep uncertainties small.

7.3.4 Fusing Poses

This section will investigate a different type of nonlinearity, the fusing of several pose estimates, as depicted in Figure 7.8. We will approach this as an estimation problem, our first involving quantities from a matrix Lie group. We will use the optimization ideas introduced in Section 7.1.9.

Figure 7.8
 Combining K pose estimates into a single fused estimate.



Theory

Suppose that we have K estimates of a pose and associated uncertainties:

$$\{\bar{\mathbf{T}}_1, \Sigma_1\}, \{\bar{\mathbf{T}}_2, \Sigma_2\}, \dots, \{\bar{\mathbf{T}}_K, \Sigma_K\}. \quad (7.311)$$

If we think of these as uncertain (pseudo)-measurements of the true pose, \mathbf{T}_{true} , how can we optimally combine these into a single estimate, $\{\bar{\mathbf{T}}, \Sigma\}$?

As we have seen in the first part of this book, vectorspace solution to fusion is straightforward and can be found exactly in closed form:

$$\bar{\mathbf{x}} = \Sigma \sum_{k=1}^K \Sigma_k^{-1} \bar{\mathbf{x}}_k, \quad \Sigma = \left(\sum_{k=1}^K \Sigma_k^{-1} \right)^{-1}. \quad (7.312)$$

The situation is somewhat more complicated when dealing with $SE(3)$, and we shall resort to an iterative scheme.

We define the error (that we will seek to minimize) as $\mathbf{e}_k(\mathbf{T})$, which occurs between the individual measurement and the optimal estimate, \mathbf{T} , so that

$$\mathbf{e}_k(\mathbf{T}) = \ln (\bar{\mathbf{T}}_k \mathbf{T}^{-1})^\vee. \quad (7.313)$$

We use our approach to pose optimization outlined earlier³⁴, wherein we start with an initial guess, \mathbf{T}_{op} , and perturb this (on the left) by a small amount, ϵ , so that

$$\mathbf{T} = \exp(\epsilon^\wedge) \mathbf{T}_{\text{op}}. \quad (7.314)$$

Inserting this into the error expression, we have

$$\begin{aligned} \mathbf{e}_k(\mathbf{T}) &= \ln (\bar{\mathbf{T}}_k \mathbf{T}^{-1})^\vee = \ln \underbrace{(\bar{\mathbf{T}}_k \mathbf{T}_{\text{op}}^{-1} \exp(-\epsilon^\wedge))^\vee}_{\text{small}} \\ &= \ln (\exp(\mathbf{e}_k(\mathbf{T}_{\text{op}})^\wedge) \exp(-\epsilon^\wedge))^\vee = \mathbf{e}_k(\mathbf{T}_{\text{op}}) - \mathbf{G}_k \epsilon, \end{aligned} \quad (7.315)$$

where $\mathbf{e}_k(\mathbf{T}_{\text{op}}) = \ln (\bar{\mathbf{T}}_k \mathbf{T}_{\text{op}}^{-1})^\vee$ and $\mathbf{G}_k = \mathcal{J}(-\mathbf{e}_k(\mathbf{T}_{\text{op}}))^{-1}$. We have used the approximate BCH formula from (7.100) to arrive at the final expression. Since $\mathbf{e}_k(\mathbf{T}_{\text{op}})$ is fairly small, this series will converge, rapidly and we can get away with keeping just a few terms. With our iterative scheme, ϵ will (hopefully) converge to zero, and hence we are justified in keeping only terms linear in this quantity.

We define the cost function that we want to minimize as

$$\begin{aligned} J(\mathbf{T}) &= \frac{1}{2} \sum_{k=1}^K \mathbf{e}_k(\mathbf{T})^T \Sigma_k^{-1} \mathbf{e}_k(\mathbf{T}) \\ &\approx \frac{1}{2} \sum_{k=1}^K (\mathbf{e}_k(\mathbf{T}_{\text{op}}) - \mathbf{G}_k \epsilon)^T \Sigma_k^{-1} (\mathbf{e}_k(\mathbf{T}_{\text{op}}) - \mathbf{G}_k \epsilon), \end{aligned} \quad (7.316)$$

which is already (approximately) quadratic in ϵ . It is in fact a squared

³⁴ It is worth mentioning that we are using our constraint-sensitive perturbations for matrix Lie groups in two distinct ways in this section. First, the perturbations are used as a means of injection noise on the Lie group so that probability and statistics can be defined. Second, we are using a perturbation to carry out iterative optimization.

Mahalanobis distance (Mahalanobis, 1936) since we have chosen the weighting matrices to be the inverse covariance matrices; thus minimizing J with respect to ϵ is equivalent to maximizing the joint likelihood of the individual estimates. It is worth noting that because we are using a constraint-sensitive perturbation scheme, we do not need to worry about enforcing any constraints on our state variables during the optimization procedure. Taking the derivative with respect to ϵ and setting to zero results in the following system of linear equations for the optimal value of ϵ :

$$\left(\sum_{k=1}^K \mathbf{G}_k^T \boldsymbol{\Sigma}_k^{-1} \mathbf{G}_k \right) \boldsymbol{\epsilon}^* = \sum_{k=1}^K \mathbf{G}_k^T \boldsymbol{\Sigma}_k^{-1} \mathbf{e}_k(\mathbf{T}_{\text{op}}). \quad (7.317)$$

While this may appear strange compared to (7.312), the Jacobian terms appear because our choice of error definition is in fact nonlinear owing to the presence of the matrix exponentials. We then apply this optimal perturbation to our current guess,

$$\mathbf{T}_{\text{op}} \leftarrow \exp(\boldsymbol{\epsilon}^{*\wedge}) \mathbf{T}_{\text{op}}, \quad (7.318)$$

which ensures \mathbf{T}_{op} remains in $SE(3)$, and iterate to convergence. At the last iteration, we take $\bar{\mathbf{T}} = \mathbf{T}_{\text{op}}$ as the mean of our fused estimate and

$$\boldsymbol{\Sigma} = \left(\sum_{k=1}^K \mathbf{G}_k^T \boldsymbol{\Sigma}_k^{-1} \mathbf{G}_k \right)^{-1} \quad (7.319)$$

for the covariance matrix. This approach has the form of a Gauss-Newton method as discussed in Section 7.1.9.

This fusion problem is similar to one investigated by Smith et al. (2003), but they only discuss the $K = 2$ case. Our approach is closer to that of Long et al. (2012), who discuss the $N = 2$ case and derive closed-form expressions for the fused mean and covariance for an arbitrary number of individual measurements, K ; however, they do not iterate their solution and they are tracking a slightly different PDF. Wolfe et al. (2011) also discuss fusion at length, albeit again using a slightly different PDF than us. They discuss non-iterative methods of fusion for arbitrary K and show numerical results for $K = 2$. We believe our approach generalizes all of these previous works by (i) allowing the number of individual estimates, K , to be arbitrary, (ii) keeping an arbitrary number of terms in the approximation of the inverse Jacobian, N , and (iii) iterating to convergence via a Gauss-Newton style optimization method. Our approach may also be simpler to implement than some of these previous methods.

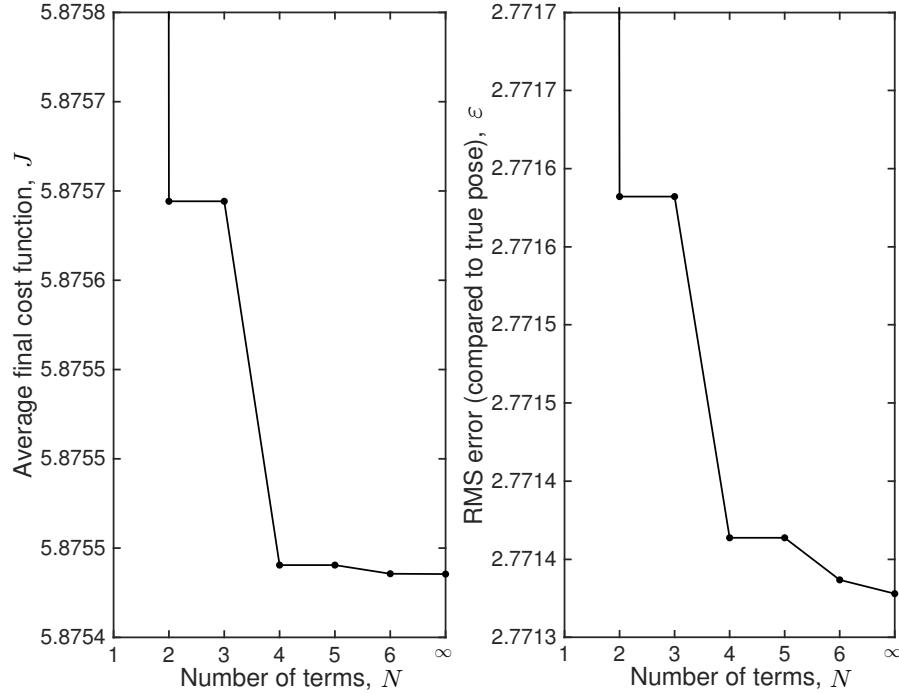


Figure 7.9
Results from Fusion Experiment: (left) average final cost function, J , as a function of the number of terms, N , kept in \mathcal{J}^{-1} ; (right) same for the root-mean-squared pose error with respect to the true pose. Both plots show that there is benefit in keeping more than one term in \mathcal{J}^{-1} . The data point denoted ‘ ∞ ’ uses the analytical expression to keep all the terms in the expansion.

Fusion Experiment

To validate the pose fusion method from the previous subsection, we used a true pose given by

$$\mathbf{T}_{\text{true}} = \exp(\boldsymbol{\xi}_{\text{true}}^\wedge), \quad \boldsymbol{\xi}_{\text{true}} = [1 \ 0 \ 0 \ 0 \ 0 \ \pi/6]^T, \quad (7.320)$$

and then generated three random pose measurements,

$$\bar{\mathbf{T}}_1 = \exp(\boldsymbol{\epsilon}_1^\wedge) \mathbf{T}_{\text{true}}, \quad \bar{\mathbf{T}}_2 = \exp(\boldsymbol{\epsilon}_2^\wedge) \mathbf{T}_{\text{true}}, \quad \bar{\mathbf{T}}_3 = \exp(\boldsymbol{\epsilon}_3^\wedge) \mathbf{T}_{\text{true}}, \quad (7.321)$$

where

$$\begin{aligned} \boldsymbol{\epsilon}_1 &\sim \mathcal{N}\left(\mathbf{0}, \text{diag}\left\{10, 5, 5, \frac{1}{2}, 1, \frac{1}{2}\right\}\right), \\ \boldsymbol{\epsilon}_2 &\sim \mathcal{N}\left(\mathbf{0}, \text{diag}\left\{5, 15, 5, \frac{1}{2}, \frac{1}{2}, 1\right\}\right), \\ \boldsymbol{\epsilon}_3 &\sim \mathcal{N}\left(\mathbf{0}, \text{diag}\left\{5, 5, 25, 1, \frac{1}{2}, \frac{1}{2}\right\}\right). \end{aligned} \quad (7.322)$$

We then solved for the pose using our Gauss-Newton technique (iterating until convergence), using the initial condition, $\mathbf{T}_{\text{op}} = \mathbf{1}$. We repeated this for $N = 1 \dots 6$, the number of terms kept in $\mathbf{G}_k = \mathcal{J}(-\mathbf{e}_k(\mathbf{T}_{\text{op}}))^{-1}$. We also used the closed-form expression to compute \mathcal{J} analytically (and then inverted numerically), and this is denoted by ‘ $N = \infty$ ’.

Figure 7.10
 Results from Fusion Experiment: (left) convergence of the cost function, J , with successive Gauss-Newton iterations. This is for just one of the M trials used to generate Figure 7.9; (right) same as left, but zoomed in to show that the $N = 2, 4, \infty$ solutions do converge to progressively lower costs.

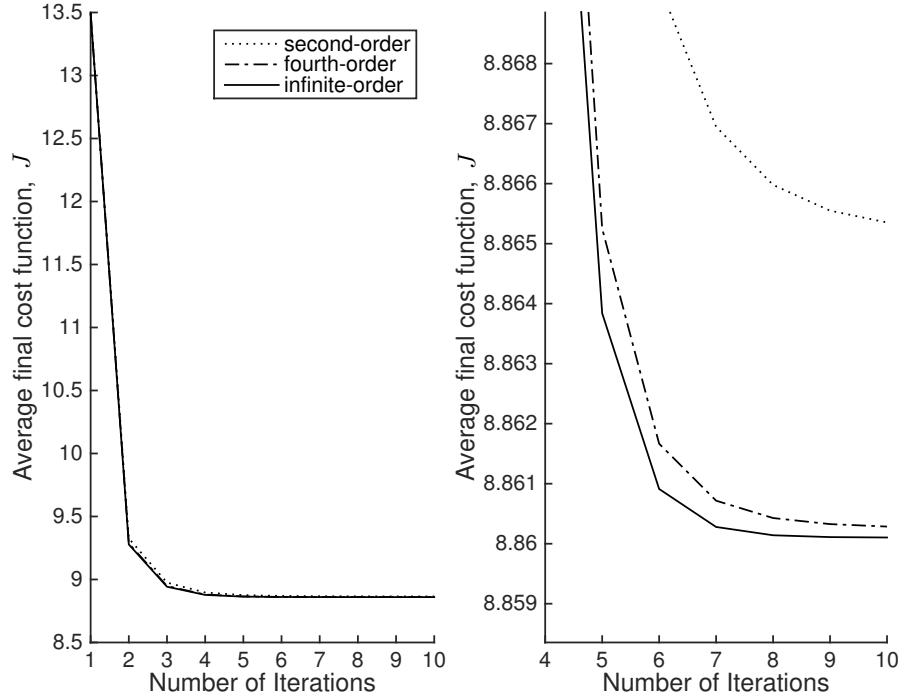


Figure 7.9 plots two performance metrics. First, it plots the final converged value of the cost function, J_m , averaged over $M = 1000$ random trials, $J = \frac{1}{M} \sum_{m=1}^M J_m$. Second, it plots the root-mean-squared pose error (with respect to the true pose), of our estimate, $\bar{\mathbf{T}}_m$, again averaged over the same M random trials:

$$\varepsilon = \sqrt{\frac{1}{M} \sum_{m=1}^M \boldsymbol{\varepsilon}_m^T \boldsymbol{\varepsilon}_m}, \quad \boldsymbol{\varepsilon}_m = \ln(\mathbf{T}_{\text{true}} \bar{\mathbf{T}}_m^{-1})^\vee.$$

The plots show that both measures of error are monotonically reduced with increasing N . Moreover, we see that for this example almost all of the benefit is gained with just four terms (or possibly even two). The results for $N = 2, 3$ are identical as are those for $N = 4, 5$. This is because in the Bernoulli number sequence, $B_3 = 0$ and $B_5 = 0$, so these terms make no additional contribution to \mathcal{J}^{-1} . It is also worth stating that if we make the rotational part of the covariances in (7.322) any bigger, we end up with a lot of samples that have rotated by more than angle π , and this can be problematic for the performance metrics we are using.

Figure 7.10 shows the convergence history of the cost, J , for a single random trial. The left side shows the strong benefit of iterating over the solution, while the right side shows that the cost converges to a lower value by keeping more terms in the approximation of \mathcal{J}^{-1} (cases

$N = 2, 4, \infty$ shown). It would seem that taking $N = 4$ for about seven iterations gains most of the benefit, for this example.

7.3.5 Propagating Uncertainty through a Nonlinear Camera Model

In estimation problems, we are often faced with passing uncertain quantities through nonlinear measurement models to produce expected measurements. Typically this is carried out via linearization (Matthies and Shafer, 1987). Sibley (2007) shows how to carry out a second-order propagation for a stereo camera model accounting for landmark uncertainty, but not pose uncertainty. Here we derive the full second-order expression for the mean (and covariance) and compare this with Monte Carlo, the sigma-point transformation, and linearization. We begin by discussing our representation of points and then present the Taylor-series expansion of the measurement (camera) model followed by an experiment.

Perturbing Homogeneous points

As we have seen in Section 7.1.8, points in \mathbb{R}^3 can be represented using 4×1 homogeneous coordinates as follows:

$$\mathbf{p} = \begin{bmatrix} sx \\ sy \\ sz \\ s \end{bmatrix}, \quad (7.323)$$

where s is some real, non-negative scalar.

To perturb points in homogeneous coordinates we will operate directly on the xyz components by writing

$$\mathbf{p} = \bar{\mathbf{p}} + \mathbf{D} \zeta, \quad (7.324)$$

where $\zeta \in \mathbb{R}^3$ is the perturbation and \mathbf{D} is a dilation matrix given by

$$\mathbf{D} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}. \quad (7.325)$$

We thus have that $E[\mathbf{p}] = \bar{\mathbf{p}}$ and

$$E[(\mathbf{p} - \bar{\mathbf{p}})(\mathbf{p} - \bar{\mathbf{p}})^T] = \mathbf{D} E[\zeta \zeta^T] \mathbf{D}^T, \quad (7.326)$$

with no approximation.

Taylor-Series Expansion of Camera Model

It is common to linearize a nonlinear observation model for use in pose estimation. In this section, we show how to do a more general Taylor-series expansion of such a model and work out the second-order case in detail. Our camera model will be

$$\mathbf{y} = \mathbf{g}(\mathbf{T}, \mathbf{p}), \quad (7.327)$$

where \mathbf{T} is the pose of the camera and \mathbf{p} is the position of a landmark (as a homogeneous point). Our task will be to pass a Gaussian representation of the pose and landmark, given by $\{\bar{\mathbf{T}}, \bar{\mathbf{p}}, \Xi\}$ where Ξ is a 9×9 covariance for both quantities, through the camera model to produce a mean and covariance for the measurement³⁵, $\{\mathbf{y}, \mathbf{R}\}$.

We can think of this as the composition of two nonlinearities, one to transfer the landmark into the camera frame, $\mathbf{z}(\mathbf{T}, \mathbf{p}) = \mathbf{T}\mathbf{p}$, and one to produce the observations from the point in the camera frame, $\mathbf{y} = \mathbf{s}(\mathbf{z})$. Thus we have

$$\mathbf{g}(\mathbf{T}, \mathbf{p}) = \mathbf{s}(\mathbf{z}(\mathbf{T}, \mathbf{p})). \quad (7.328)$$

We will treat each one in turn. If we change the pose of the camera and/or the position of the landmark a little bit, we have

$$\mathbf{z} = \mathbf{T}\mathbf{p} = \exp(\boldsymbol{\epsilon}^\wedge) \bar{\mathbf{T}} (\bar{\mathbf{p}} + \mathbf{D} \boldsymbol{\zeta}) \approx \left(\mathbf{1} + \boldsymbol{\epsilon}^\wedge + \frac{1}{2} \boldsymbol{\epsilon}^\wedge \boldsymbol{\epsilon}^\wedge \right) \bar{\mathbf{T}} (\bar{\mathbf{p}} + \mathbf{D} \boldsymbol{\zeta}), \quad (7.329)$$

where we have kept the first two terms in the Taylor series for the pose perturbation. If we multiply out and continue to keep only those terms that are second-order or lower in $\boldsymbol{\epsilon}$ and $\boldsymbol{\zeta}$ we have

$$\mathbf{z} \approx \bar{\mathbf{z}} + \mathbf{Z} \boldsymbol{\theta} + \frac{1}{2} \sum_{i=1}^4 \underbrace{\boldsymbol{\theta}^T \mathcal{Z}_i \boldsymbol{\theta}}_{\text{scalar}} \mathbf{1}_i, \quad (7.330)$$

where $\mathbf{1}_i$ is the i th column of the 4×4 identity matrix and

$$\bar{\mathbf{z}} = \bar{\mathbf{T}} \bar{\mathbf{p}}, \quad (7.331a)$$

$$\mathbf{Z} = \begin{bmatrix} (\bar{\mathbf{T}} \bar{\mathbf{p}})^\odot & \bar{\mathbf{T}} \mathbf{D} \end{bmatrix}, \quad (7.331b)$$

$$\mathcal{Z}_i = \begin{bmatrix} \mathbf{1}_i^\odot (\bar{\mathbf{T}} \bar{\mathbf{p}})^\odot & \mathbf{1}_i^\odot \bar{\mathbf{T}} \mathbf{D} \\ (\mathbf{1}_i^\odot \bar{\mathbf{T}} \mathbf{D})^T & \mathbf{0} \end{bmatrix}, \quad (7.331c)$$

$$\boldsymbol{\theta} = \begin{bmatrix} \boldsymbol{\epsilon} \\ \boldsymbol{\zeta} \end{bmatrix}. \quad (7.331d)$$

Arriving at these expressions requires repeated application of the identities from Section 7.1.8.

³⁵ In this example, the only sources of uncertainty come from the pose and the point and we neglect inherent measurement noise, but this could be incorporated as additive Gaussian noise, if desired.

To then apply the nonlinear camera model, we use the chain rule (for first and second derivatives), so that

$$\mathbf{g}(\mathbf{T}, \mathbf{p}) \approx \bar{\mathbf{g}} + \mathbf{G}\boldsymbol{\theta} + \frac{1}{2} \sum_j \underbrace{\boldsymbol{\theta}^T \mathcal{G}_j \boldsymbol{\theta}}_{\text{scalar}} \mathbf{1}_j, \quad (7.332)$$

correct to second order in $\boldsymbol{\theta}$, where

$$\bar{\mathbf{g}} = \mathbf{s}(\bar{\mathbf{z}}), \quad (7.333a)$$

$$\mathbf{G} = \mathbf{S}\mathbf{Z}, \quad \mathbf{S} = \left. \frac{\partial \mathbf{s}}{\partial \mathbf{z}} \right|_{\bar{\mathbf{z}}}, \quad (7.333b)$$

$$\mathcal{G}_j = \mathbf{Z}^T \mathcal{S}_j \mathbf{Z} + \sum_{i=1}^4 \underbrace{\mathbf{1}_j^T \mathbf{S} \mathbf{1}_i}_{\text{scalar}} \mathcal{Z}_i, \quad (7.333c)$$

$$\mathcal{S}_j = \left. \frac{\partial^2 s_j}{\partial \mathbf{z} \partial \mathbf{z}^T} \right|_{\bar{\mathbf{z}}}, \quad (7.333d)$$

j is an index over the rows of $\mathbf{s}(\cdot)$, and $\mathbf{1}_j$ is the j th column of the identity matrix. The Jacobian of $\mathbf{s}(\cdot)$ is \mathbf{S} and the Hessian of the j th row, $s_j(\cdot)$, is \mathcal{S}_j .

If we only care about the first-order perturbation, we simply have

$$\mathbf{g}(\mathbf{T}, \mathbf{p}) = \bar{\mathbf{g}} + \mathbf{G}\boldsymbol{\theta}, \quad (7.334)$$

where $\bar{\mathbf{g}}$ and \mathbf{G} are unchanged from above.

These perturbed measurement equations can then be used within any estimation scheme we like; in the next subsection we will use these with a stereo camera model to show the benefit of the second-order terms.

Propagating Gaussian Uncertainty through the Camera

Suppose that the input uncertainties, embodied by $\boldsymbol{\theta}$, are zero-mean, Gaussian,

$$\boldsymbol{\theta} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Xi}), \quad (7.335)$$

where we note that in general there could be correlations between the pose, \mathbf{T} , and the landmark, \mathbf{p} .

Then, to first order, our measurement is given by

$$\mathbf{y}_{1\text{st}} = \bar{\mathbf{g}} + \mathbf{G}\boldsymbol{\theta}, \quad (7.336)$$

and $\bar{\mathbf{y}}_{1\text{st}} = E[\mathbf{y}_{1\text{st}}] = \bar{\mathbf{g}}$ since $E[\boldsymbol{\theta}] = \mathbf{0}$ by assumption. The (second-order) covariance associated with the first-order camera model is given by

$$\mathbf{R}_{2\text{nd}} = E \left[(\mathbf{y}_{1\text{st}} - \bar{\mathbf{y}}_{1\text{st}}) (\mathbf{y}_{1\text{st}} - \bar{\mathbf{y}}_{1\text{st}})^T \right] = \mathbf{G}\boldsymbol{\Xi}\mathbf{G}^T. \quad (7.337)$$

For the second-order camera model, we have

$$\mathbf{y}_{\text{2nd}} = \bar{\mathbf{g}} + \mathbf{G} \boldsymbol{\theta} + \frac{1}{2} \sum_j \boldsymbol{\theta}^T \mathcal{G}_j \boldsymbol{\theta} \mathbf{1}_j, \quad (7.338)$$

and consequently,

$$\bar{\mathbf{y}}_{\text{2nd}} = E[\mathbf{y}_{\text{2nd}}] = \bar{\mathbf{g}} + \frac{1}{2} \sum_j \text{tr}(\mathcal{G}_j \boldsymbol{\Xi}) \mathbf{1}_j, \quad (7.339)$$

which has an extra non-zero term as compared to the first-order camera model. The larger the input covariance $\boldsymbol{\Xi}$ is, the larger this term can become, depending on the nonlinearity. For a linear camera model, $\mathcal{G}_j = \mathbf{0}$ and the second- and first-order camera model means are identical.

We will also compute a (fourth-order) covariance, but with just second-order terms in the camera model expansion. To do this properly, we should expand the camera model to third order as there is an additional fourth-order covariance term involving the product of first- and third-order camera-model terms; however, this would involve a complicated expression employing the third derivative of the camera model. As such, the approximate fourth-order covariance we will use is given by

$$\begin{aligned} \mathbf{R}_{\text{4th}} &\approx E \left[(\mathbf{y}_{\text{2nd}} - \bar{\mathbf{y}}_{\text{2nd}}) (\mathbf{y}_{\text{2nd}} - \bar{\mathbf{y}}_{\text{2nd}})^T \right] \\ &= \mathbf{G} \boldsymbol{\Xi} \mathbf{G}^T - \frac{1}{4} \left(\sum_{i=1}^J \text{tr}(\mathcal{G}_i \boldsymbol{\Xi}) \mathbf{1}_i \right) \left(\sum_{j=1}^J \text{tr}(\mathcal{G}_j \boldsymbol{\Xi}) \mathbf{1}_j \right)^T \\ &\quad + \frac{1}{4} \sum_{i,j=1}^J \sum_{k,\ell,m,n=1}^9 \mathcal{G}_{ik\ell} \mathcal{G}_{jm\ell} \left(\boldsymbol{\Xi}_{k\ell} \boldsymbol{\Xi}_{mn} + \boldsymbol{\Xi}_{km} \boldsymbol{\Xi}_{\ell n} + \boldsymbol{\Xi}_{kn} \boldsymbol{\Xi}_{\ell m} \right), \end{aligned} \quad (7.340)$$

where $\mathcal{G}_{ik\ell}$ is the $k\ell$ th element of \mathcal{G}_i and $\boldsymbol{\Xi}_{k\ell}$ is the $k\ell$ th element of $\boldsymbol{\Xi}$. The first- and third-order terms in the covariance expansion are identically zero owing to the symmetry of the Gaussian density. The last term in the above makes use of *Isserlis' theorem* for Gaussian variables.

Sigmapoint Method

Finally, we can also make use of the sigmapoint transformation to pass uncertainty through the nonlinear camera model. As in the pose compounding problem, we tailor this to our specific type of $SE(3)$ perturbation. We begin by approximating the input Gaussian using a finite

number of samples, $\{\mathbf{T}_\ell, \mathbf{p}_\ell\}$:

$$\mathbf{L}\mathbf{L}^T = \boldsymbol{\Xi}, \quad (\text{Cholesky decomposition; } \mathbf{L} \text{ lower-triangular}) \quad (7.341\text{a})$$

$$\boldsymbol{\theta}_\ell = \mathbf{0}, \quad (7.341\text{b})$$

$$\boldsymbol{\theta}_\ell = \sqrt{L + \kappa} \operatorname{col}_\ell \mathbf{L}, \quad \ell = 1 \dots L, \quad (7.341\text{c})$$

$$\boldsymbol{\theta}_{\ell+L} = -\sqrt{L + \kappa} \operatorname{col}_\ell \mathbf{L}, \quad \ell = 1 \dots L, \quad (7.341\text{d})$$

$$\begin{bmatrix} \boldsymbol{\epsilon}_\ell \\ \boldsymbol{\zeta}_\ell \end{bmatrix} = \boldsymbol{\theta}_\ell, \quad (7.341\text{e})$$

$$\mathbf{T}_\ell = \exp(\hat{\boldsymbol{\epsilon}}_\ell) \bar{\mathbf{T}}, \quad (7.341\text{f})$$

$$\mathbf{p}_\ell = \bar{\mathbf{p}} + \mathbf{D} \boldsymbol{\zeta}_\ell, \quad (7.341\text{g})$$

where κ is a user-definable constant³⁶ and $L = 9$. We then pass each of these samples through the nonlinear camera model:

$$\mathbf{y}_\ell = \mathbf{s}(\mathbf{T}_\ell \mathbf{p}_\ell), \quad \ell = 0 \dots 2L. \quad (7.342)$$

These are combined to create the output mean and covariance according to

$$\bar{\mathbf{y}}_{\text{sp}} = \frac{1}{L + \kappa} \left(\kappa \mathbf{y}_0 + \frac{1}{2} \sum_{\ell=1}^{2L} \mathbf{y}_\ell \right), \quad (7.343\text{a})$$

$$\begin{aligned} \mathbf{R}_{\text{sp}} = \frac{1}{L + \kappa} & \left(\kappa (\mathbf{y}_0 - \bar{\mathbf{y}}_{\text{sp}})(\mathbf{y}_0 - \bar{\mathbf{y}}_{\text{sp}})^T \right. \\ & \left. + \frac{1}{2} \sum_{\ell=1}^{2L} (\mathbf{y}_\ell - \bar{\mathbf{y}}_{\text{sp}})(\mathbf{y}_\ell - \bar{\mathbf{y}}_{\text{sp}})^T \right). \quad (7.343\text{b}) \end{aligned}$$

The next section will provide the details for a specific nonlinear camera model, $\mathbf{s}(\cdot)$, representing a stereo camera.

Stereo Camera Model

To demonstrate the propagation of uncertainty through a nonlinear measurement model, $\mathbf{s}(\cdot)$, we will employ our midpoint stereo camera model given by

$$\mathbf{s}(\boldsymbol{\rho}) = \mathbf{M} \frac{1}{z_3} \mathbf{z}, \quad (7.344)$$

where

$$\mathbf{s} = \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \end{bmatrix}, \quad \mathbf{z} = \begin{bmatrix} \boldsymbol{\rho} \\ 1 \end{bmatrix} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{bmatrix}, \quad \mathbf{M} = \begin{bmatrix} f_u & 0 & c_u & f_u \frac{b}{2} \\ 0 & f_v & c_v & 0 \\ f_u & 0 & c_u & -f_u \frac{b}{2} \\ 0 & f_v & c_v & 0 \end{bmatrix}, \quad (7.345)$$

and f_u, f_v are the horizontal, vertical focal lengths (in pixels), (c_u, c_v) is the optical center of the images (in pixels), and b is the separation

³⁶ For all experiments in this section, we used $\kappa = 0$.

between the cameras (in metres). The optical axis of the camera is along the z_3 , direction.

The Jacobian of this measurement model is given by

$$\frac{\partial \mathbf{s}}{\partial \mathbf{z}} = \mathbf{M} \frac{1}{z_3} \begin{bmatrix} 1 & 0 & -\frac{z_1}{z_3} & 0 \\ 0 & 1 & -\frac{z_2}{z_3} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{z_4}{z_3} & 1 \end{bmatrix}, \quad (7.346)$$

and the Hessian is given by

$$\begin{aligned} \frac{\partial^2 s_1}{\partial \mathbf{z} \partial \mathbf{z}^T} &= \frac{f_u}{z_3^2} \begin{bmatrix} 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & \frac{2z_1+bz_4}{z_3} & -\frac{b}{2} \\ 0 & 0 & -\frac{b}{2} & 0 \end{bmatrix}, \\ \frac{\partial^2 s_2}{\partial \mathbf{z} \partial \mathbf{z}^T} &= \frac{\partial^2 s_4}{\partial \mathbf{z} \partial \mathbf{z}^T} = \frac{f_v}{z_3^2} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & -1 & \frac{2z_2}{z_3} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \\ \frac{\partial^2 s_3}{\partial \mathbf{z} \partial \mathbf{z}^T} &= \frac{f_u}{z_3^2} \begin{bmatrix} 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & \frac{2z_1-bz_4}{z_3} & \frac{b}{2} \\ 0 & 0 & \frac{b}{2} & 0 \end{bmatrix}, \end{aligned} \quad (7.347)$$

where we have shown each component separately.

Camera Experiment

We used the following methods to pass a Gaussian uncertainty on camera pose and landmark position through the nonlinear stereo camera model:

- (i) *Monte Carlo*: We drew a large number, $M = 1,000,000$, of random samples from the input density, passed these through the camera model, and then computed the mean, $\bar{\mathbf{y}}_{mc}$, and covariance, \mathbf{R}_{mc} . This slow-but-accurate approach served as our benchmark to which the other three much faster methods were compared.
- (ii) *First/Second-Order*: We used the first-order camera model to compute $\bar{\mathbf{y}}_{1st}$ and \mathbf{R}_{2nd} , as described above.
- (iii) *Second/Fourth-Order*: We used the second-order camera model to compute $\bar{\mathbf{y}}_{2nd}$ and \mathbf{R}_{4th} , as described above.
- (iv) *Sigmapoint*: We used the sigmapoint method described above to compute $\bar{\mathbf{y}}_{sp}$ and \mathbf{R}_{sp} .

The camera parameters were

$$b = 0.25 \text{ m}, \quad f_u = f_v = 200 \text{ pixels}, \quad c_u = c_v = 0 \text{ pixels}.$$

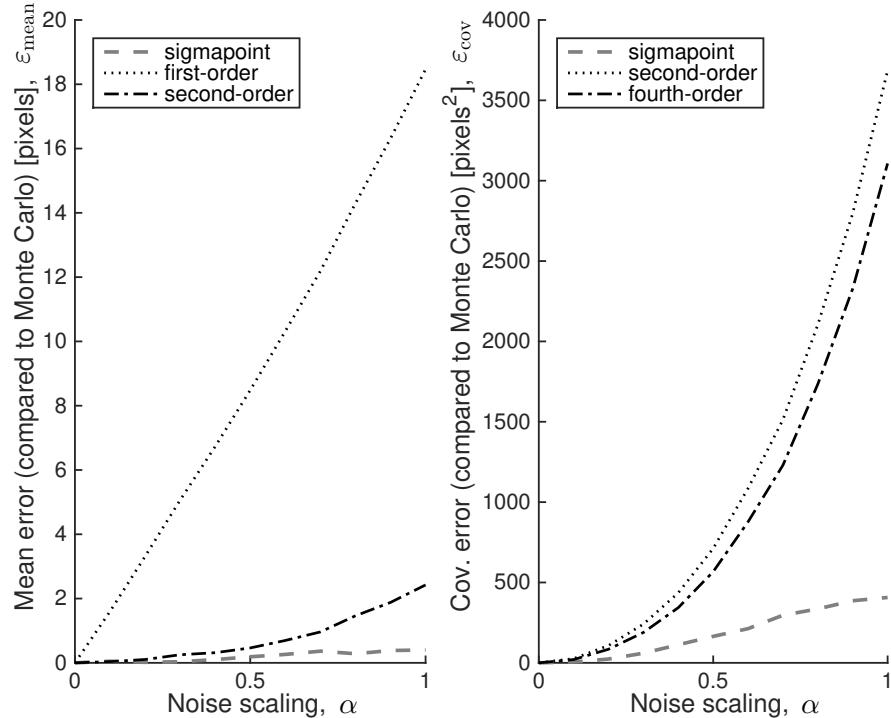


Figure 7.11
Results from Stereo Camera Experiment: (left) mean and (right) covariance errors, $\varepsilon_{\text{mean}}$ and ε_{cov} , for three methods of passing a Gaussian uncertainty through a nonlinear stereo camera model, as compared to Monte Carlo. The parameter, α , scales the magnitude of the input covariance matrix.

We used the camera pose $\mathbf{T} = \mathbf{1}$ and let the landmark be located at $\mathbf{p} = [10 \ 10 \ 10 \ 1]^T$. For the combined pose/landmark uncertainty, we used an input covariance of

$$\mathbf{\Sigma} = \alpha \times \text{diag} \left\{ \frac{1}{10}, \frac{1}{10}, \frac{1}{10}, \frac{1}{100}, \frac{1}{100}, \frac{1}{100}, 1, 1, 1 \right\},$$

where $\alpha \in [0, 1]$ is a scaling parameter that allowed us to parametrically increase the magnitude of the uncertainty.

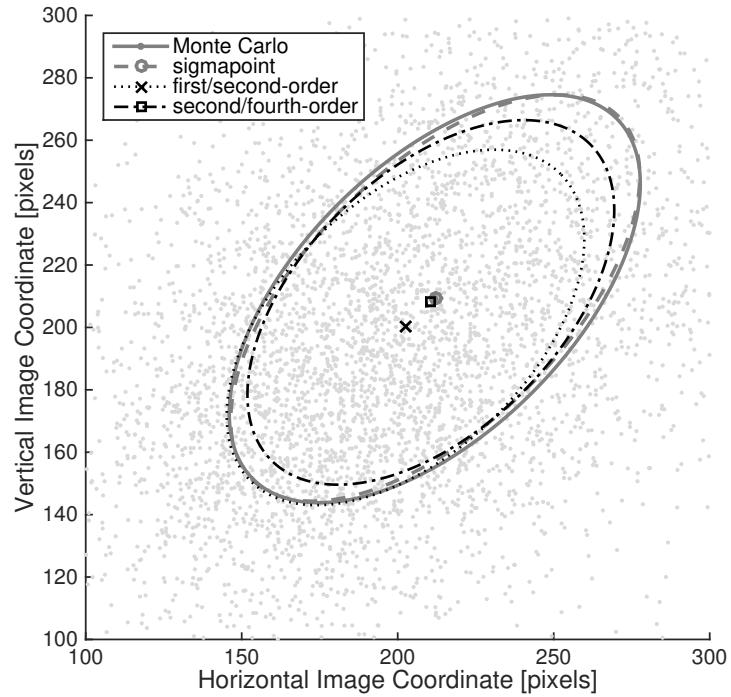
To gauge performance, we evaluated both the mean and covariance of each method by comparing the results to those of the Monte Carlo simulation according to the following metrics:

$$\begin{aligned} \varepsilon_{\text{mean}} &= \sqrt{(\bar{\mathbf{y}} - \bar{\mathbf{y}}_{\text{mc}})^T (\bar{\mathbf{y}} - \bar{\mathbf{y}}_{\text{mc}})}, \\ \varepsilon_{\text{cov}} &= \sqrt{\text{tr}((\mathbf{R} - \mathbf{R}_{\text{mc}})^T (\mathbf{R} - \mathbf{R}_{\text{mc}}))}, \end{aligned}$$

where the latter is the *Frobenius norm*.

Figure 7.11 shows the two performance metrics, $\varepsilon_{\text{mean}}$ and ε_{cov} , for each of the three techniques over a wide range of noise scalings, α . We see that the sigmapoint technique does the best on both mean and covariance. The second-order technique does reasonably well on the mean, but the corresponding fourth-order technique does poorly

Figure 7.12
 Results from Stereo Camera Experiment: A portion of the left image of a stereo camera showing the mean and covariance (as a one-standard-deviation ellipse) for four methods of imaging a landmark with Gaussian uncertainty on the camera's pose and the landmark's position. This case corresponds to the $\alpha = 1$ data point in Figure 7.11.



on the covariance (due to our inability to compute a fully fourth-order-accurate covariance, as explained earlier).

Figure 7.12 provides a snapshot of a portion of the left image of the stereo camera with the mean and one-standard-deviation covariance ellipses shown for all techniques. We see that the sigmapoint technique does an excellent job on both the mean and the covariance, while the others do not fare as well.

7.4 Summary

The main take-away points from this chapter are as follows:

1. While rotations and poses cannot be described using vectorspaces, we can describe them using the matrix Lie groups, $SO(3)$ and $SE(3)$.
2. We can perturb both rotations and poses conveniently by using the exponential map, which (surjective-only) maps \mathbb{R}^3 and \mathbb{R}^6 to $SO(3)$ and $SE(3)$, respectively. We can use this mapping for two different purposes within state estimation:
 - (i) to adjust a point estimate (i.e., mean or MAP) of rotation or pose by a little bit during an optimal estimation procedure.
 - (ii) to define Gaussian-like PDFs for rotations and poses by mapping Gaussian noise onto $SO(3)$ and $SE(3)$ through the exponential map.

3. Our ability to represent uncertainty for rotations and poses using the methods in this chapter is limited to only small amounts. We cannot represent uncertainty globally on $SO(3)$ and $SE(3)$ using our Gaussian-like PDFs; for this, refer to Chirikjian and Kyatkin (2016). However, these methods are good enough to allow us to modify the estimation techniques from the first part of the book for use with rotations and poses.

The last part of the book will bring together these matrix-Lie-group tools with the estimation techniques from the first part of the book, in order to carry out state estimation for practical robotics problems.

7.5 Exercises

7.5.1 Prove that

$$(\mathbf{C}\mathbf{u})^\wedge \equiv \mathbf{C}\mathbf{u}^\wedge \mathbf{C}^T.$$

7.5.2 Prove that

$$(\mathbf{C}\mathbf{u})^\wedge \equiv (2 \cos \phi + 1)\mathbf{u}^\wedge - \mathbf{u}^\wedge \mathbf{C} - \mathbf{C}^T \mathbf{u}^\wedge.$$

7.5.3 Prove that

$$\exp((\mathbf{C}\mathbf{u})^\wedge) \equiv \mathbf{C} \exp(\mathbf{u}^\wedge) \mathbf{C}^T.$$

7.5.4 Prove that

$$(\mathcal{T}\mathbf{x})^\wedge \equiv \mathbf{T}\mathbf{x}^\wedge \mathbf{T}^{-1}.$$

7.5.5 Prove that

$$\exp((\mathcal{T}\mathbf{x})^\wedge) \equiv \mathbf{T} \exp(\mathbf{x}^\wedge) \mathbf{T}^{-1}.$$

7.5.6 Work out the expression for $\mathbf{Q}_\ell(\boldsymbol{\xi})$ in (7.86b).

7.5.7 Prove that

$$\mathbf{x}^\wedge \mathbf{p} \equiv \mathbf{p}^\odot \mathbf{x}.$$

7.5.8 Prove that

$$\mathbf{p}^T \mathbf{x}^\wedge \equiv \mathbf{x}^T \mathbf{p}^\odot.$$

7.5.9 Prove that

$$\int_0^1 \alpha^n (1-\alpha)^m d\alpha \equiv \frac{n! m!}{(n+m+1)!}.$$

Hint: use integration by parts.

7.5.10 Prove the identity

$$\dot{\mathbf{J}}(\boldsymbol{\phi}) - \boldsymbol{\omega}^\wedge \mathbf{J}(\boldsymbol{\phi}) \equiv \frac{\partial \boldsymbol{\omega}}{\partial \boldsymbol{\phi}},$$

where

$$\boldsymbol{\omega} = \mathbf{J}(\boldsymbol{\phi}) \dot{\boldsymbol{\phi}},$$

are the rotational kinematics expressed in $\mathfrak{so}(3)$. Hint: it can be shown one term at a time by writing out each quantity as a series.

7.5.11 Show that

$$(\mathbf{T}\mathbf{p})^\odot \equiv \mathbf{T}\mathbf{p}^\odot \boldsymbol{\mathcal{T}}^{-1}.$$

7.5.12 Show that

$$(\mathbf{T}\mathbf{p})^{\odot^T} (\mathbf{T}\mathbf{p})^\odot \equiv \boldsymbol{\mathcal{T}}^{-T} \mathbf{p}^{\odot^T} \mathbf{p}^\odot \boldsymbol{\mathcal{T}}^{-1}.$$

7.5.13 Starting from the $SE(3)$ kinematics,

$$\dot{\mathbf{T}} = \boldsymbol{\varpi}^\wedge \mathbf{T},$$

show that the kinematics can also be written using the adjoint quantities:

$$\dot{\boldsymbol{\mathcal{T}}} = \boldsymbol{\varpi}^\wedge \boldsymbol{\mathcal{T}}.$$

7.5.14 Show that it is possible to work with a modified version of the homogeneous-point representation when using the adjoint quantities:

$$\underbrace{\text{Ad}(\overbrace{\mathbf{T}\mathbf{p}}^{4 \times 1})}_{6 \times 3} = \underbrace{\text{Ad}(\mathbf{T})}_{6 \times 6} \underbrace{\text{Ad}(\mathbf{p})}_{6 \times 3},$$

where we abuse notation and define an adjoint operator for a homogeneous point as

$$\begin{aligned} \text{Ad} \left(\begin{bmatrix} \mathbf{c} \\ 1 \end{bmatrix} \right) &= \begin{bmatrix} \mathbf{c}^\wedge \\ \mathbf{1} \end{bmatrix}, \\ \text{Ad}^{-1} \left(\begin{bmatrix} \mathbf{A} \\ \mathbf{B} \end{bmatrix} \right) &= \begin{bmatrix} (\mathbf{A}\mathbf{B}^{-1})^\vee \\ 1 \end{bmatrix}, \end{aligned}$$

with \mathbf{c} a 3×1 and \mathbf{A}, \mathbf{B} both 3×3 .

Part III

Applications

8

Pose Estimation Problems

In this last part of the book, we will address some key three-dimensional estimation problems from robotics. We will bring together the ideas from Part I on classic state estimation with the three-dimensional machinery of Part II.

This chapter will start by looking at a key problem, aligning two point-clouds (i.e., collections of points) using the principle of least squares. We will then return to the EKF and batch state estimators and adjust these to work with rotation and pose variables, in the context of a specific pose estimation problem. Our focus will be on localization of a vehicle when the geometry of the world is known. The next chapter will address the more difficult scenario of unknown world geometry.

8.1 Point-Cloud Alignment

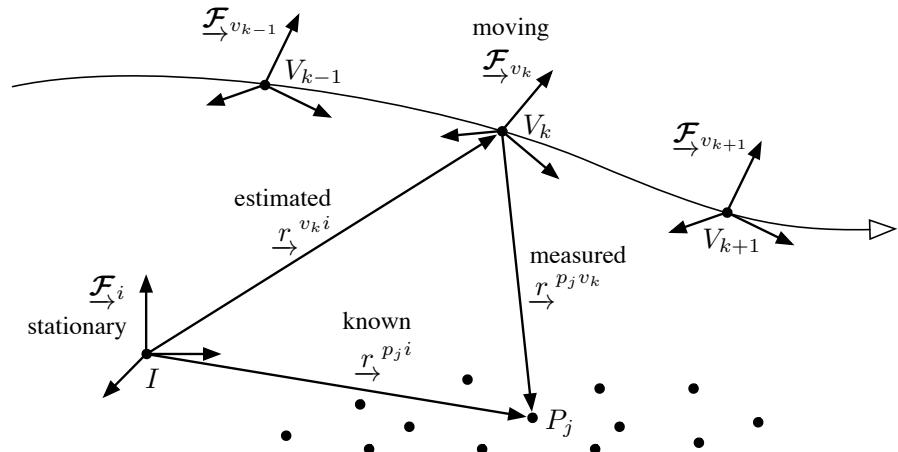
In this section, we will study a classic result in pose estimation. Specifically, we present the solution for aligning two sets of three-dimensional points, or *point-clouds*, while minimizing a least-squares cost function. The caveat is that the weights associated with each term in the cost function must be *scalars*, not matrices; this can be referred to as *ordinary* least squares¹.

This result is used commonly in the popular *iterative closest point (ICP)* (Besl and McKay, 1992) algorithm for aligning three-dimensional points to a three-dimensional model. It is also used inside outlier rejection schemes, such as RANSAC (Fischler and Bolles, 1981) (see Section 5.3.1), for rapid pose determination using a minimal set of points.

We will present the solution using three different parameterizations of the rotation/pose variable: unit-length quaternions, rotation matrices, and then transformation matrices. The (non-iterative) quaternion approach comes down to solving an eigenproblem, while the (non-iterative) rotation-matrix approach turns into a singular-value decomposition. Finally, the iterative transformation matrix approach only involves solving a system of linear equations.

¹ This problem finds its origin in spacecraft attitude determination, with the famous *Wahba's problem* (Wahba, 1965).

Figure 8.1
 Definition of reference frames for a point-cloud alignment problem. There is a stationary reference frame and a moving reference frame, attached to a vehicle. A collection of points, P_j , is observed in both frames, and the goal is to determine the relative pose of the moving frame with respect to the stationary one by aligning the two point-clouds.



8.1.1 Problem Setup

We will use the setup in Figure 8.1. There are two reference frames, one non-moving, \mathcal{F}_i , and one attached to a moving vehicle, \mathcal{F}_{v_k} . In particular, we have M measurements, $\mathbf{r}_{v_k}^{p_j v_k}$, where $j = 1 \dots M$, of points from the vehicle (expressed in the moving frame \mathcal{F}_{v_k}). We assume these measurements could have been corrupted by noise.

Let us assume that we know $\mathbf{r}_i^{p_j i}$, the position of each point, P_j , located and expressed in the non-moving frame, \mathcal{F}_i . For example, in the ICP algorithm, these points are determined by finding the closest point on the model to each observed point. Thus, we seek to align a collection of M points expressed in two different references frames. In other words, we want to find the translation and rotation that best align the two point-clouds². Note that in this first problem we are only carrying out the alignment at a single time, k . We will consider a point-cloud tracking problem later in the chapter.

8.1.2 Unit-Length Quaternion Solution

We will present the unit-length quaternion approach to aligning point-clouds first³. This solution was first studied by Davenport (1965) in the aerospace world and later by Horn (1987b) in robotics. We will use our quaternion notation defined earlier in Section 6.2.3. The quaternion approach has an advantage over the rotation-matrix case (to be described in the next section) because the constraints required to produce a valid rotation are easier for unit-length quaternions.

² An unknown scale between the two point-clouds is also sometimes folded into the problem; we will assume the two point-clouds have the same scale.

³ Our focus in this book is on the use of rotation matrices, but this is an example of a problem where unit-length quaternions make things easier, and therefore we include the derivation.

To work with quaternions, we define the following 4×1 homogeneous versions of our points:

$$\mathbf{y}_j = \begin{bmatrix} \mathbf{r}_{v_k}^{p_j v_k} \\ 1 \end{bmatrix}, \quad \mathbf{p}_j = \begin{bmatrix} \mathbf{r}_i^{p_j i} \\ 1 \end{bmatrix}, \quad (8.1)$$

where we have dropped the sub- and superscripts, except for j , the point index.

We would like to find the translation, \mathbf{r} , and rotation, \mathbf{q} , that best align these points, thereby giving us the relative pose between $\underline{\mathcal{F}}_{v_k}$ and $\underline{\mathcal{F}}_i$. We note that the relationships between the quaternion versions of the translation, \mathbf{r} , and rotation, \mathbf{q} , and our usual 3×1 translation, $\mathbf{r}_i^{v_k i}$, and 3×3 rotation matrix, $\mathbf{C}_{v_k i}$ are defined by

$$\underbrace{\begin{bmatrix} \mathbf{r}_{v_k}^{p_j v_k} \\ 1 \end{bmatrix}}_{\mathbf{y}_j} = \underbrace{\begin{bmatrix} \mathbf{C}_{v_k i} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix}}_{\mathbf{q}^{-1+}\mathbf{q}^\oplus} \left(\underbrace{\begin{bmatrix} \mathbf{r}_i^{p_j i} \\ 1 \end{bmatrix}}_{\mathbf{p}_j} - \underbrace{\begin{bmatrix} \mathbf{r}_{v_k i} \\ 0 \end{bmatrix}}_{\mathbf{r}} \right), \quad (8.2)$$

which is just an expression of the geometry of the problem in the absence of any noise corrupting the measurements. Using the identity in (6.19), we can rewrite this as

$$\mathbf{y}_j = \mathbf{q}^{-1+}(\mathbf{p}_j - \mathbf{r})^+\mathbf{q}, \quad (8.3)$$

which again is in the absence of any noise.

Referring to (8.3), we could form an error quaternion for point P_j as

$$\mathbf{e}_j = \mathbf{y}_j - \mathbf{q}^{-1+}(\mathbf{p}_j - \mathbf{r})^+\mathbf{q}, \quad (8.4)$$

but instead we can manipulate the above to generate an error that appears linear in \mathbf{q} :

$$\mathbf{e}'_j = \mathbf{q}^+ \mathbf{e}_j = (\mathbf{y}_j^\oplus - (\mathbf{p}_j - \mathbf{r})^+) \mathbf{q}. \quad (8.5)$$

We will define the total objective function (to minimize), J , as

$$J(\mathbf{q}, \mathbf{r}, \lambda) = \frac{1}{2} \sum_{j=1}^M w_j \mathbf{e}'_j^T \mathbf{e}'_j - \underbrace{\frac{1}{2} \lambda (\mathbf{q}^T \mathbf{q} - 1)}_{\text{Lagrange multiplier term}}, \quad (8.6)$$

where the w_j are unique scalar weights assigned to each of the point pairs. We have included the Lagrange multiplier term on the right to ensure the unit-length constraint on the rotation quaternion. It is also worth noting that selecting \mathbf{e}'_j over \mathbf{e}_j has no effect on our objective function since

$$\mathbf{e}'_j^T \mathbf{e}'_j = (\mathbf{q}^+ \mathbf{e}_j)^T (\mathbf{q}^+ \mathbf{e}_j) = \mathbf{e}_j^T \mathbf{q}^{+T} \mathbf{q}^+ \mathbf{e}_j = \mathbf{e}_j^T (\mathbf{q}^{-1+} \mathbf{q})^+ \mathbf{e}_j = \mathbf{e}_j^T \mathbf{e}_j. \quad (8.7)$$

Inserting the expression for \mathbf{e}'_j into the objective function, we see

$$\begin{aligned} J(\mathbf{q}, \mathbf{r}, \lambda) &= \frac{1}{2} \sum_{j=1}^M w_j \mathbf{q}^T \left(\mathbf{y}_j^\oplus - (\mathbf{p}_j - \mathbf{r})^+ \right)^T \left(\mathbf{y}_j^\oplus - (\mathbf{p}_j - \mathbf{r})^+ \right) \mathbf{q} \\ &\quad - \frac{1}{2} \lambda (\mathbf{q}^T \mathbf{q} - 1). \end{aligned} \quad (8.8)$$

Taking the derivative of the objective function with respect to \mathbf{q} , \mathbf{r} , and λ , we find

$$\frac{\partial J}{\partial \mathbf{q}^T} = \sum_{j=1}^M w_j \left(\mathbf{y}_j^\oplus - (\mathbf{p}_j - \mathbf{r})^+ \right)^T \left(\mathbf{y}_j^\oplus - (\mathbf{p}_j - \mathbf{r})^+ \right) \mathbf{q} - \lambda \mathbf{q}, \quad (8.9a)$$

$$\frac{\partial J}{\partial \mathbf{r}^T} = \mathbf{q}^{-1\oplus} \sum_{j=1}^M w_j \left(\mathbf{y}_j^\oplus - (\mathbf{p}_j - \mathbf{r})^+ \right) \mathbf{q}, \quad (8.9b)$$

$$\frac{\partial J}{\partial \lambda} = -\frac{1}{2} (\mathbf{q}^T \mathbf{q} - 1). \quad (8.9c)$$

Setting the second to zero, we find

$$\mathbf{r} = \mathbf{p} - \mathbf{q}^+ \mathbf{y}^+ \mathbf{q}^{-1}, \quad (8.10)$$

where \mathbf{p} and \mathbf{y} are defined below. Thus, the optimal translation is the difference of the centroids of the two point-clouds, in the stationary frame.

Substituting \mathbf{r} into the first and setting to zero, we can show

$$\mathbf{W}\mathbf{q} = \lambda \mathbf{q}, \quad (8.11)$$

where

$$\mathbf{W} = \frac{1}{w} \sum_{j=1}^M w_j \left((\mathbf{y}_j - \mathbf{y})^\oplus - (\mathbf{p}_j - \mathbf{p})^+ \right)^T \left((\mathbf{y}_j - \mathbf{y})^\oplus - (\mathbf{p}_j - \mathbf{p})^+ \right), \quad (8.12a)$$

$$\mathbf{y} = \frac{1}{w} \sum_{j=1}^M w_j \mathbf{y}_j, \quad \mathbf{p} = \frac{1}{w} \sum_{j=1}^M w_j \mathbf{p}_j, \quad w = \sum_{j=1}^M w_j. \quad (8.12b)$$

We can see this is just an eigenproblem⁴. If the eigenvalues are positive and the smallest eigenvalue is distinct (i.e., not repeated), then finding the smallest eigenvalue and the corresponding unique eigenvector will yield \mathbf{q} to within a multiplicative constant and our constraint that $\mathbf{q}^T \mathbf{q} = 1$ makes the solution unique.

⁴ The *eigenproblem* associated with an $N \times N$ matrix, \mathbf{A} , is defined by the equation $\mathbf{Ax} = \lambda \mathbf{x}$. The N (not necessarily distinct) *eigenvalues*, λ_i , are found by solving for the roots of $\det(\mathbf{A} - \lambda \mathbf{1}) = 0$ and then for each eigenvalue the corresponding *eigenvector*, \mathbf{x}_i , is found (to within a multiplicative constant) through substitution of the eigenvalue into the original equation and appropriate manipulation. The case of non-distinct eigenvalues is tricky and requires advanced linear algebra.

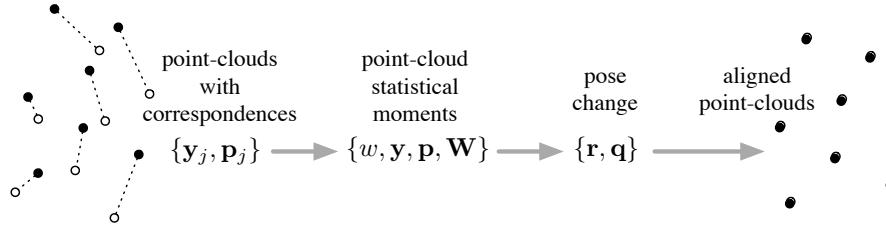


Figure 8.2 Steps involved in aligning two point-clouds.

To see that we want the smallest eigenvalue, we first note that \mathbf{W} is both symmetric and positive-semidefinite. Positive-semidefiniteness implies that all the eigenvalues of \mathbf{W} are non-negative. Next, we can set (8.9a) to zero so that an equivalent expression for \mathbf{W} is

$$\mathbf{W} = \sum_{j=1}^M w_j \left(\mathbf{y}_j^\oplus - (\mathbf{p}_j - \mathbf{r})^+ \right)^T \left(\mathbf{y}_j^\oplus - (\mathbf{p}_j - \mathbf{r})^+ \right). \quad (8.13)$$

Substituting this into the objective function in (8.8), we immediately see that

$$J(\mathbf{q}, \mathbf{r}, \lambda) = \frac{1}{2} \mathbf{q}^T \underbrace{\mathbf{W} \mathbf{q}}_{\lambda \mathbf{q}} - \frac{1}{2} \lambda (\mathbf{q}^T \mathbf{q} - 1) = \frac{1}{2} \lambda. \quad (8.14)$$

Thus, picking the smallest possible value for λ will minimize the objective function.

However, there are some complications if \mathbf{W} is singular or the smallest eigenvalue is not distinct. Then there can be multiple choices for the eigenvector corresponding to the smallest eigenvalue, and therefore the solution may not be unique. We will forgo discussing this further for the quaternion method as it would require advanced linear algebra techniques (e.g., Jordan normal form) and instead return to this issue in the next section when using rotation matrices.

Note, we have not made any approximations or linearizations in our technique, but this depends heavily on the fact that the weights are scalar not matrices. Figure 8.2 shows the process to align two point-clouds. Once we have the \mathbf{r} and \mathbf{q} , we can construct the final estimates of the rotation matrix, $\hat{\mathbf{C}}_{v_k i}$, and translation, $\hat{\mathbf{r}}_i^{v_k i}$, from

$$\begin{bmatrix} \hat{\mathbf{C}}_{v_k i} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} = \mathbf{q}^{-1+} \mathbf{q}^\oplus, \quad \begin{bmatrix} \hat{\mathbf{r}}_i^{v_k i} \\ 0 \end{bmatrix} = \mathbf{r}, \quad (8.15)$$

and then we can construct an estimated transformation matrix according to

$$\hat{\mathbf{T}}_{v_k i} = \begin{bmatrix} \hat{\mathbf{C}}_{v_k i} & -\hat{\mathbf{C}}_{v_k i} \hat{\mathbf{r}}_i^{v_k i} \\ \mathbf{0}^T & 1 \end{bmatrix}, \quad (8.16)$$

which combines our rotation and translation into a single answer for

the best alignment of the point-clouds. Referring back to Section 6.3.2, we may actually be interested in $\hat{\mathbf{T}}_{iv_k}$, which can be recovered using

$$\hat{\mathbf{T}}_{iv_k} = \hat{\mathbf{T}}_{v_k i}^{-1} = \begin{bmatrix} \hat{\mathbf{C}}_{iv_k} & \hat{\mathbf{r}}_i^{v_k i} \\ \mathbf{0}^T & 1 \end{bmatrix}. \quad (8.17)$$

Both forms of the transformation matrix are useful, depending on how the solution will be used.

8.1.3 Rotation Matrix Solution

The rotation-matrix case was originally studied outside of robotics by Green (1952) and Wahba (1965) and later within robotics by Horn (1987a) and Arun et al. (1987) and later by Umeyama (1991) considering the $\det \mathbf{C} = 1$ constraint. We follow the approach of de Ruiter and Forbes (2013), which captures all of the cases in which \mathbf{C} can be determined uniquely. We also identify how many global and local solutions can exist for \mathbf{C} when there is not a single global solution.

As in the previous section, we will use some simplified notation to avoid repeating sub- and super-scripts:

$$\mathbf{y}_j = \mathbf{r}_{v_k}^{p_j v_k}, \quad \mathbf{p}_j = \mathbf{r}_i^{p_j i}, \quad \mathbf{r} = \mathbf{r}_i^{v_k i}, \quad \mathbf{C} = \mathbf{C}_{v_k i}. \quad (8.18)$$

Also, we define

$$\mathbf{y} = \frac{1}{w} \sum_{j=1}^M w_j \mathbf{y}_j, \quad \mathbf{p} = \frac{1}{w} \sum_{j=1}^M w_j \mathbf{p}_j, \quad w = \sum_{j=1}^M w_j, \quad (8.19)$$

where the w_j are scalar weights for each point. Note that, as compared to the last section, some of the symbols are now 3×1 rather than 4×1 .

We define an error term for each point:

$$\mathbf{e}_j = \mathbf{y}_j - \mathbf{C}(\mathbf{p}_j - \mathbf{r}). \quad (8.20)$$

Our estimation problem is then to globally minimize the cost function,

$$J(\mathbf{C}, \mathbf{r}) = \frac{1}{2} \sum_{j=1}^M w_j \mathbf{e}_j^T \mathbf{e}_j = \frac{1}{2} \sum_{j=1}^M w_j (\mathbf{y}_j - \mathbf{C}(\mathbf{p}_j - \mathbf{r}))^T (\mathbf{y}_j - \mathbf{C}(\mathbf{p}_j - \mathbf{r})), \quad (8.21)$$

subject to $\mathbf{C} \in SO(3)$ (i.e., $\mathbf{C}\mathbf{C}^T = \mathbf{1}$ and $\det \mathbf{C} = 1$).

Before carrying out the optimization, we will make a change of variables for the translation parameter. Define

$$\mathbf{d} = \mathbf{r} + \mathbf{C}^T \mathbf{y} - \mathbf{p}, \quad (8.22)$$

which is easy to isolate for \mathbf{r} if all the other quantities are known. In

this case, we can rewrite our cost function as

$$\begin{aligned} J(\mathbf{C}, \mathbf{d}) = & \underbrace{\frac{1}{2} \sum_{j=1}^M w_j ((\mathbf{y}_j - \mathbf{y}) - \mathbf{C}(\mathbf{p}_j - \mathbf{p}))^T ((\mathbf{y}_j - \mathbf{y}) - \mathbf{C}(\mathbf{p}_j - \mathbf{p}))}_{\text{depends only on } \mathbf{C}} \\ & + \underbrace{\frac{1}{2} \mathbf{d}^T \mathbf{d}}_{\text{depends only on } \mathbf{d}}, \quad (8.23) \end{aligned}$$

which is the sum of two semi-positive-definite terms, the first depending only on \mathbf{C} and the second only on \mathbf{d} . We can minimize the second trivially by taking $\mathbf{d} = \mathbf{0}$, which in turn implies that

$$\mathbf{r} = \mathbf{p} - \mathbf{C}^T \mathbf{y}. \quad (8.24)$$

As in the quaternion case, this is simply the difference of the centroids of the two point-clouds, expressed in the stationary frame.

What is left is to minimize the first term with respect to \mathbf{C} . We note that if we multiply out each smaller term within the first large term, only one part actually depends on \mathbf{C} :

$$\begin{aligned} & ((\mathbf{y}_j - \mathbf{y}) - \mathbf{C}(\mathbf{p}_j - \mathbf{p}))^T ((\mathbf{y}_j - \mathbf{y}) - \mathbf{C}(\mathbf{p}_j - \mathbf{p})) \\ = & \underbrace{(\mathbf{y}_j - \mathbf{y})^T (\mathbf{y}_j - \mathbf{y})}_{\text{independent of } \mathbf{C}} - 2 \underbrace{((\mathbf{y}_j - \mathbf{y})^T \mathbf{C}(\mathbf{p}_j - \mathbf{p}))}_{\text{tr}(\mathbf{C}(\mathbf{p}_j - \mathbf{p})(\mathbf{y}_j - \mathbf{y})^T)} + \underbrace{(\mathbf{p}_j - \mathbf{p})^T (\mathbf{p}_j - \mathbf{p})}_{\text{independent of } \mathbf{C}}. \quad (8.25) \end{aligned}$$

Summing this middle term over all the (weighted) points, we have

$$\begin{aligned} \frac{1}{w} \sum_{j=1}^M w_j ((\mathbf{y}_j - \mathbf{y})^T \mathbf{C}(\mathbf{p}_j - \mathbf{p})) &= \frac{1}{w} \sum_{j=1}^M w_j \text{tr}(\mathbf{C}(\mathbf{p}_j - \mathbf{p})(\mathbf{y}_j - \mathbf{y})^T) \\ &= \text{tr} \left(\mathbf{C} \frac{1}{w} \sum_{j=1}^M w_j (\mathbf{p}_j - \mathbf{p})(\mathbf{y}_j - \mathbf{y})^T \right) = \text{tr}(\mathbf{CW}^T), \quad (8.26) \end{aligned}$$

where

$$\mathbf{W} = \frac{1}{w} \sum_{j=1}^M w_j (\mathbf{y}_j - \mathbf{y})(\mathbf{p}_j - \mathbf{p})^T. \quad (8.27)$$

This \mathbf{W} matrix plays a similar role to the one in the quaternion section, by capturing the spread of the points (similar to an inertia matrix in dynamics), but it is not exactly the same. Therefore, we can define a new cost function that we seek to minimize with respect to \mathbf{C} as

$$J(\mathbf{C}, \boldsymbol{\Lambda}, \gamma) = -\text{tr}(\mathbf{CW}^T) + \underbrace{\text{tr}(\boldsymbol{\Lambda}(\mathbf{CC}^T - \mathbf{1}))}_{\text{Lagrange multiplier terms}} + \gamma(\det \mathbf{C} - 1), \quad (8.28)$$

where $\boldsymbol{\Lambda}$ and γ are Lagrange multipliers associated with the two terms

on the right; these are used to ensure that the resulting $\mathbf{C} \in SO(3)$. Note that when $\mathbf{CC}^T = \mathbf{1}$ and $\det \mathbf{C} = 1$, these terms have no effect on the resulting cost. It is also worth noting that $\mathbf{\Lambda}$ is symmetric since we only need to enforce six orthogonality constraints. This new cost function will be minimized by the same \mathbf{C} as our original one.

Taking the derivative of $J(\mathbf{C}, \mathbf{\Lambda}, \gamma)$ with respect to \mathbf{C} , $\mathbf{\Lambda}$, and γ , we have⁵

$$\frac{\partial J}{\partial \mathbf{C}} = -\mathbf{W} + 2\mathbf{\Lambda}\mathbf{C} + \gamma \underbrace{\det \mathbf{C}}_1 \underbrace{\mathbf{C}^{-T}}_{\mathbf{C}} = -\mathbf{W} + \mathbf{LC}, \quad (8.29a)$$

$$\frac{\partial J}{\partial \mathbf{\Lambda}} = \mathbf{CC}^T - \mathbf{1}, \quad (8.29b)$$

$$\frac{\partial J}{\partial \gamma} = \det \mathbf{C} - 1, \quad (8.29c)$$

where we have lumped together the Lagrange multipliers as $\mathbf{L} = 2\mathbf{\Lambda} + \gamma\mathbf{1}$. Setting the first equation to zero, we find that

$$\mathbf{LC} = \mathbf{W}. \quad (8.30)$$

At this point, our explanation can proceed in a simplified or detailed manner, depending on the level of fidelity we want to capture.

Before moving forward, we show that it is possible to arrive at (8.30) using our Lie group tools without the use of Lagrange multipliers. We consider a perturbation of the rotation matrix of the form

$$\mathbf{C}' = \exp(\phi^\wedge) \mathbf{C}, \quad (8.31)$$

and then take the derivative of the objective function with respect to ϕ and set this to zero for a critical point. For the derivative with respect to the i th element of ϕ we have

$$\begin{aligned} \frac{\partial J}{\partial \phi_i} &= \lim_{h \rightarrow 0} \frac{J(\mathbf{C}') - J(\mathbf{C})}{h} \\ &= \lim_{h \rightarrow 0} \frac{-\text{tr}(\mathbf{C}'\mathbf{W}^T) + \text{tr}(\mathbf{CW}^T)}{h} \\ &= \lim_{h \rightarrow 0} \frac{-\text{tr}(\exp(h\mathbf{1}_i^\wedge)\mathbf{CW}^T) + \text{tr}(\mathbf{CW}^T)}{h} \\ &\approx \lim_{h \rightarrow 0} \frac{-\text{tr}((\mathbf{1} + h\mathbf{1}_i^\wedge)\mathbf{CW}^T) + \text{tr}(\mathbf{CW}^T)}{h} \\ &= \lim_{h \rightarrow 0} \frac{-\text{tr}(h\mathbf{1}_i^\wedge \mathbf{CW}^T)}{h} \\ &= -\text{tr}(\mathbf{1}_i^\wedge \mathbf{CW}^T). \end{aligned} \quad (8.32)$$

⁵ We require these useful facts to take the derivatives:

$$\begin{aligned} \frac{\partial}{\partial \mathbf{A}} \det \mathbf{A} &= \det(\mathbf{A}) \mathbf{A}^{-T}, \\ \frac{\partial}{\partial \mathbf{A}} \text{tr}(\mathbf{AB}^T) &= \mathbf{B}, \\ \frac{\partial}{\partial \mathbf{A}} \text{tr}(\mathbf{BA}\mathbf{A}^T) &= (\mathbf{B} + \mathbf{B}^T)\mathbf{A}. \end{aligned}$$

Setting this to zero, we require

$$(\forall i) \operatorname{tr} \left(\mathbf{1}_i^\wedge \underbrace{\mathbf{C} \mathbf{W}^T}_{\mathbf{L}} \right) = 0. \quad (8.33)$$

Owing to the skew-symmetric nature of the \wedge operator, this implies that $\mathbf{L} = \mathbf{C} \mathbf{W}^T$ is a symmetric matrix for a critical point. Taking the transpose and right-multiplying by \mathbf{C} , we come back to (8.30). We now continue with the main derivation.

Simplified Explanation

If we somehow knew that $\det \mathbf{W} > 0$, then we could proceed as follows. First, we postmultiply (8.30) by itself transposed to find

$$\underbrace{\mathbf{L} \mathbf{C} \mathbf{C}^T \mathbf{L}^T}_{\mathbf{1}} = \mathbf{W} \mathbf{W}^T. \quad (8.34)$$

Since \mathbf{L} is symmetric, we have that

$$\mathbf{L} = (\mathbf{W} \mathbf{W}^T)^{\frac{1}{2}}, \quad (8.35)$$

which we see involves a matrix square-root. Substituting this back into (8.30), the optimal rotation is

$$\mathbf{C} = (\mathbf{W} \mathbf{W}^T)^{-\frac{1}{2}} \mathbf{W}. \quad (8.36)$$

This has the same form as the projection onto $SO(3)$ discussed in Section 7.2.1.

Unfortunately, this approach does not tell the entire story since it relies on assuming something about \mathbf{W} , and therefore does not capture all of the subtleties of the problem. With lots of non-coplanar points, this method will typically work well. However, there are some difficult cases for which we need a more detailed analysis. A common situation in which this problem occurs is when carrying out alignments using just three pairs of noisy points in the RANSAC algorithm discussed earlier. The next section provides a more thorough analysis of the solution that handles the difficult cases.

Detailed Explanation

The detailed explanation begins by first carrying out a singular-value decomposition (SVD)⁶ on the (square, real) matrix, \mathbf{W} , so that

$$\mathbf{W} = \mathbf{U} \mathbf{D} \mathbf{V}^T, \quad (8.37)$$

⁶ The *singular-value decomposition* of a real $M \times N$ matrix, \mathbf{A} , is a factorization of the form $\mathbf{A} = \mathbf{U} \mathbf{D} \mathbf{V}^T$ where \mathbf{U} is an $M \times M$ real, orthogonal matrix (i.e., $\mathbf{U}^T \mathbf{U} = \mathbf{1}$), \mathbf{D} is an $M \times N$ matrix with real entries $d_i \geq 0$ on the main diagonal (all other entries zero), and \mathbf{V} is an $N \times N$ real, orthogonal matrix (i.e., $\mathbf{V}^T \mathbf{V} = \mathbf{1}$). The d_i are called the *singular values* and are typically ordered from largest to smallest along the diagonal of \mathbf{D} . Note that the SVD is not unique.

where \mathbf{U} and \mathbf{V} are square, orthogonal matrices and $\mathbf{D} = \text{diag}(d_1, d_2, d_3)$ is a diagonal matrix of singular values, $d_1 \geq d_2 \geq d_3 \geq 0$.

Returning to (8.30), we can substitute in the SVD of \mathbf{W} so that

$$\mathbf{L}^2 = \mathbf{LL}^T = \mathbf{LCC}^T\mathbf{L}^T = \mathbf{WW}^T = \mathbf{UD}\underbrace{\mathbf{V}^T\mathbf{V}}_1\mathbf{D}^T\mathbf{U}^T = \mathbf{UD}^2\mathbf{U}^T. \quad (8.38)$$

Taking the matrix square-root, we can write that

$$\mathbf{L} = \mathbf{UMU}^T, \quad (8.39)$$

where \mathbf{M} is the symmetric, matrix square root of \mathbf{D}^2 . In other words,

$$\mathbf{M}^2 = \mathbf{D}^2. \quad (8.40)$$

It can be shown (de Ruiter and Forbes, 2013) that every real, symmetric \mathbf{M} satisfying this condition can be written in the form

$$\mathbf{M} = \mathbf{YDSY}^T, \quad (8.41)$$

where $\mathbf{S} = \text{diag}(s_1, s_2, s_3)$ with $s_i = \pm 1$ and \mathbf{Y} an orthogonal matrix (i.e., $\mathbf{Y}^T\mathbf{Y} = \mathbf{YY}^T = \mathbf{1}$). An obvious example of this is $\mathbf{Y} = \mathbf{1}$ with $s_i = \pm 1$ and any values for d_i ; a less obvious example that is a possibility when $d_1 = d_2$ is

$$\begin{aligned} \mathbf{M} &= \begin{bmatrix} d_1 \cos \theta & d_1 \sin \theta & 0 \\ d_1 \sin \theta & -d_1 \cos \theta & 0 \\ 0 & 0 & d_3 \end{bmatrix} \\ &= \underbrace{\begin{bmatrix} \cos \frac{\theta}{2} & -\sin \frac{\theta}{2} & 0 \\ \sin \frac{\theta}{2} & \cos \frac{\theta}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{Y}} \underbrace{\begin{bmatrix} d_1 & 0 & 0 \\ 0 & -d_1 & 0 \\ 0 & 0 & d_3 \end{bmatrix}}_{\mathbf{DS}} \underbrace{\begin{bmatrix} \cos \frac{\theta}{2} & -\sin \frac{\theta}{2} & 0 \\ \sin \frac{\theta}{2} & \cos \frac{\theta}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}^T}_{\mathbf{Y}^T}, \end{aligned} \quad (8.42)$$

for any value of the free parameter, θ . This illustrates an important point, that the structure of \mathbf{Y} can become more complex in correspondence with repeated singular values (i.e., we cannot just pick any \mathbf{Y}). Related to this, we always have that

$$\mathbf{D} = \mathbf{YDY}^T, \quad (8.43)$$

due to the relationship between the block structure of \mathbf{Y} and the multiplicity of the singular values in \mathbf{D} .

Now, we can manipulate the objective function that we want to minimize as follows:

$$\begin{aligned} J &= -\text{tr}(\mathbf{CW}^T) = -\text{tr}(\mathbf{WC}^T) = -\text{tr}(\mathbf{L}) = -\text{tr}(\mathbf{UYDSY}^T\mathbf{U}^T) \\ &= -\text{tr}(\underbrace{\mathbf{Y}^T\mathbf{U}^T\mathbf{UY}}_1\mathbf{DS}) = -\text{tr}(\mathbf{DS}) = -(d_1 s_1 + d_2 s_2 + d_3 s_3). \end{aligned} \quad (8.44)$$

There are now several cases to consider.

Case (i): $\det \mathbf{W} \neq 0$

Here we have that all of the singular values are positive. From (8.30) and (8.39) we have that

$$\begin{aligned}\det \mathbf{W} &= \det \mathbf{L} \underbrace{\det \mathbf{C}}_1 = \det \mathbf{L} = \det(\mathbf{U} \mathbf{Y} \mathbf{D} \mathbf{S} \mathbf{Y}^T \mathbf{U}^T) \\ &= \underbrace{\det(\mathbf{Y}^T \mathbf{U}^T \mathbf{U} \mathbf{Y})}_1 \det \mathbf{D} \det \mathbf{S} = \underbrace{\det \mathbf{D}}_{>0} \det \mathbf{S}. \quad (8.45)\end{aligned}$$

Since the singular values are positive, we have that $\det \mathbf{D} > 0$. Or in other words, the signs of the determinants of \mathbf{S} and \mathbf{W} must be the same, which implies that

$$\begin{aligned}\det \mathbf{S} &= \text{sgn}(\det \mathbf{S}) = \text{sgn}(\det \mathbf{W}) = \text{sgn}(\det(\mathbf{U} \mathbf{D} \mathbf{V}^T)) \\ &= \text{sgn}(\underbrace{\det \mathbf{U}}_{\pm 1} \underbrace{\det \mathbf{D}}_{>0} \underbrace{\det \mathbf{V}}_{\pm 1}) = \det \mathbf{U} \det \mathbf{V} = \pm 1. \quad (8.46)\end{aligned}$$

Note that we have $\det \mathbf{U} = \pm 1$ since $(\det \mathbf{U})^2 = \det(\mathbf{U}^T \mathbf{U}) = \det \mathbf{1} = 1$ and the same for \mathbf{V} . There are now four subcases to consider:

Subcase (i-a): $\det \mathbf{W} > 0$

Since $\det \mathbf{W} > 0$ by assumption, we must also have $\det \mathbf{S} = 1$ and therefore to uniquely minimize J in (8.44) we must pick $s_1 = s_2 = s_3 = 1$ since all of the d_i are positive and therefore we must have \mathbf{Y} diagonal. Thus, from (8.30) we have

$$\begin{aligned}\mathbf{C} &= \mathbf{L}^{-1} \mathbf{W} = (\mathbf{U} \mathbf{Y} \mathbf{D} \mathbf{S} \mathbf{Y}^T \mathbf{U}^T)^{-1} \mathbf{U} \mathbf{D} \mathbf{V}^T \\ &= \mathbf{U} \mathbf{Y} \underbrace{\mathbf{S}^{-1} \mathbf{D}^{-1} \mathbf{Y}^T}_{\mathbf{S}} \underbrace{\mathbf{U}^T \mathbf{U}}_1 \mathbf{D} \mathbf{V}^T = \mathbf{U} \mathbf{Y} \mathbf{S} \mathbf{D}^{-1} \underbrace{\mathbf{Y}^T \mathbf{D}}_{\mathbf{D} \mathbf{Y}^T} \mathbf{V}^T \\ &= \mathbf{U} \mathbf{Y} \mathbf{S} \mathbf{Y}^T \mathbf{V}^T = \mathbf{U} \mathbf{S} \mathbf{V}^T, \quad (8.47)\end{aligned}$$

with $\mathbf{S} = \text{diag}(1, 1, 1) = \mathbf{1}$, which is equivalent to the solution provided in our ‘simplified explanation’ in the last section.

Subcase (i-b): $\det \mathbf{W} < 0, d_1 \geq d_2 > d_3 > 0$

Since $\det \mathbf{W} < 0$ by assumption, we have $\det \mathbf{S} = -1$, which means exactly one of the s_i must be negative. In this case, we can uniquely minimize J in (8.44) since the minimum singular value, d_3 , is distinct, whereupon we must pick $s_1 = s_2 = 1$ and $s_3 = -1$ for the minimum. Since $s_1 = s_2 = 1$, we must have \mathbf{Y} diagonal and can therefore from (8.30) we have that

$$\mathbf{C} = \mathbf{U} \mathbf{S} \mathbf{V}^T, \quad (8.48)$$

with $\mathbf{S} = \text{diag}(1, 1, -1)$.

Subcase (i-c): $\det \mathbf{W} < 0$, $d_1 > d_2 = d_3 > 0$

As in the last subcase, we have $\det \mathbf{S} = -1$, which means exactly one of the s_i must be negative. Looking to (8.44), since $d_2 = d_3$ we can pick either $s_2 = -1$ or $s_3 = -1$ and end up with the same value for J . With these values for the s_i we can pick any of the following for \mathbf{Y} :

$$\mathbf{Y} = \text{diag}(\pm 1, \pm 1, \pm 1), \quad \mathbf{Y} = \begin{bmatrix} \pm 1 & 0 & 0 \\ 0 & \pm \cos \frac{\theta}{2} & \mp \sin \frac{\theta}{2} \\ 0 & \pm \sin \frac{\theta}{2} & \pm \cos \frac{\theta}{2} \end{bmatrix}, \quad (8.49)$$

where θ is a free parameter. We can plug any of these \mathbf{Y} in to find minimizing solutions for \mathbf{C} using (8.30):

$$\mathbf{C} = \mathbf{U} \mathbf{Y} \mathbf{S} \mathbf{Y}^T \mathbf{V}^T, \quad (8.50)$$

with $\mathbf{S} = \text{diag}(1, 1, -1)$ or $\mathbf{S} = \text{diag}(1, -1, 1)$. Since θ can be anything, this means there are an infinite number of solutions that minimize the objective function.

Subcase (i-d): $\det \mathbf{W} < 0$, $d_1 = d_2 = d_3 > 0$

As in the last subcase, we have $\det \mathbf{S} = -1$, which means exactly one of the s_i must be negative. Looking to (8.44), since $d_1 = d_2 = d_3$ we can pick $s_1 = -1$ or $s_2 = -1$ or $s_3 = -1$ and end up with the same value for J , implying there are an infinite number of minimizing solutions.

Case (ii): $\det \mathbf{W} = 0$

This time there are three subcases to consider depending on how many singular values are zero.

Subcase (ii-a): $\text{rank } \mathbf{W} = 2$

In this case, we have $d_1 \geq d_2 > d_3 = 0$. Looking back to (8.44) we see that we can uniquely minimize J by picking $s_1 = s_2 = 1$ and since $d_3 = 0$, the value of s_3 does not affect J and thus it is a free parameter. Again looking to (8.30) we have

$$(\mathbf{U} \mathbf{Y} \mathbf{D} \mathbf{S} \mathbf{Y}^T \mathbf{U}^T) \mathbf{C} = \mathbf{U} \mathbf{D} \mathbf{V}^T. \quad (8.51)$$

Multiplying by \mathbf{U}^T from the left and \mathbf{V} from the right, we have

$$\mathbf{D} \underbrace{\mathbf{U}^T \mathbf{C} \mathbf{V}}_{\mathbf{Q}} = \mathbf{D}, \quad (8.52)$$

since $\mathbf{D} \mathbf{S} = \mathbf{D}$ due to $d_3 = 0$ and then $\mathbf{Y} \mathbf{D} \mathbf{Y}^T = \mathbf{D}$ from (8.43). The matrix, \mathbf{Q} , above will be orthogonal since \mathbf{U} , \mathbf{C} , and \mathbf{V} are all orthogonal. Since $\mathbf{D} \mathbf{Q} = \mathbf{D}$, $\mathbf{D} = \text{diag}(d_1, d_2, 0)$, and $\mathbf{Q} \mathbf{Q}^T = \mathbf{1}$, we know that $\mathbf{Q} = \text{diag}(1, 1, q_3)$ with $q_3 = \pm 1$. We also have that

$$q_3 = \det \mathbf{Q} = \det \mathbf{U} \underbrace{\det \mathbf{C}}_1 \det \mathbf{V} = \det \mathbf{U} \det \mathbf{V} = \pm 1, \quad (8.53)$$

and therefore rearranging (and renaming \mathbf{Q} as \mathbf{S}), we have

$$\mathbf{C} = \mathbf{U}\mathbf{S}\mathbf{V}^T, \quad (8.54)$$

with $\mathbf{S} = \text{diag}(1, 1, \det \mathbf{U} \det \mathbf{V})$.

Subcase (ii-b): rank $\mathbf{W} = 1$

In this case, we have $d_1 > d_2 = d_3 = 0$. We let $s_1 = 1$ to minimize J and now s_2 and s_3 do not affect J and are free parameters. Similarly to the last subcase, we end up with an equation of the form

$$\mathbf{D}\mathbf{Q} = \mathbf{D}, \quad (8.55)$$

which, along with $\mathbf{D} = \text{diag}(d_1, 0, 0)$ and $\mathbf{Q}\mathbf{Q}^T = \mathbf{1}$, implies that \mathbf{Q} will have one of the following forms:

$$\underbrace{\mathbf{Q} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta \\ 0 & \sin \theta & \cos \theta \end{bmatrix}}_{\det \mathbf{Q}=1} \quad \text{or} \quad \underbrace{\mathbf{Q} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & \sin \theta \\ 0 & \sin \theta & -\cos \theta \end{bmatrix}}_{\det \mathbf{Q}=-1}, \quad (8.56)$$

with $\theta \in \mathbb{R}$ a free parameter. This means there are infinitely many minimizing solutions. Since

$$\det \mathbf{Q} = \det \mathbf{U} \underbrace{\det \mathbf{C}}_1 \det \mathbf{V} = \det \mathbf{U} \det \mathbf{V} = \pm 1, \quad (8.57)$$

we have (renaming \mathbf{Q} as \mathbf{S}) that

$$\mathbf{C} = \mathbf{U}\mathbf{S}\mathbf{V}^T, \quad (8.58)$$

with

$$\mathbf{S} = \begin{cases} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta \\ 0 & \sin \theta & \cos \theta \end{bmatrix} & \text{if } \det \mathbf{U} \det \mathbf{V} = 1 \\ \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & \sin \theta \\ 0 & \sin \theta & -\cos \theta \end{bmatrix} & \text{if } \det \mathbf{U} \det \mathbf{V} = -1 \end{cases}. \quad (8.59)$$

Physically, this case corresponds to all of the points being collinear (in at least one of the frames) so that rotating about the axis formed by the points through any angle, θ , does not alter the objective function J .

Subcase (ii-c): rank $\mathbf{W} = 0$

This case corresponds to there being no points or all the points coincident and so any $\mathbf{C} \in SO(3)$ will produce the same value of the objective function, J .

Summary:

We have provided all of the solutions for \mathbf{C} in our point-alignment problem; depending on the properties of \mathbf{W} , there can be one or infinitely many global solutions. Looking back through all the cases and subcases, we can see that if there is a unique global solution for \mathbf{C} , it is always of the form

$$\mathbf{C} = \mathbf{USV}^T, \quad (8.60)$$

with $\mathbf{S} = \text{diag}(1, 1, \det \mathbf{U} \det \mathbf{V})$ and $\mathbf{W} = \mathbf{UDV}^T$ is a singular-value decomposition of \mathbf{W} . The necessary and sufficient conditions for this unique global solution to exist are:

- (i) $\det \mathbf{W} > 0$, or
- (ii) $\det \mathbf{W} < 0$ and minimum singular value distinct: $d_1 \geq d_2 > d_3 > 0$, or
- (iii) $\text{rank } \mathbf{W} = 2$.

If none of these conditions is true, there will be infinite solutions for \mathbf{C} . However, these cases are fairly pathological and do not occur frequently in practical situations.

Once we have solved for the optimal rotation matrix, we take $\hat{\mathbf{C}}_{v_k i} = \mathbf{C}$ as our estimated rotation. We build the estimated translation as

$$\hat{\mathbf{r}}_i^{v_k i} = \mathbf{p} - \hat{\mathbf{C}}_{v_k i}^T \mathbf{y} \quad (8.61)$$

and, if desired, combine the translation and rotation into an estimated transformation matrix,

$$\hat{\mathbf{T}}_{v_k i} = \begin{bmatrix} \hat{\mathbf{C}}_{v_k i} & -\hat{\mathbf{C}}_{v_k i} \hat{\mathbf{r}}_i^{v_k i} \\ \mathbf{0}^T & 1 \end{bmatrix}, \quad (8.62)$$

that provides the optimal alignment of the two point-clouds in a single quantity. Again, as mentioned in Section 6.3.2, we may actually be interested in $\hat{\mathbf{T}}_{iv_k}$, which can be recovered using

$$\hat{\mathbf{T}}_{iv_k} = \hat{\mathbf{T}}_{v_k i}^{-1} = \begin{bmatrix} \hat{\mathbf{C}}_{iv_k} & \hat{\mathbf{r}}_i^{v_k i} \\ \mathbf{0}^T & 1 \end{bmatrix}. \quad (8.63)$$

Both forms of the transformation matrix are useful, depending on how the solution will be used.

Example 8.1 We provide an example of *subcase (i-b)* to make things tangible. Consider the following two point-clouds that we wish to align, each consisting of six points:

$$\begin{aligned} \mathbf{p}_1 &= 3 \times \mathbf{1}_1, \quad \mathbf{p}_2 = 2 \times \mathbf{1}_2, \quad \mathbf{p}_3 = \mathbf{1}_3, \quad \mathbf{p}_4 = -3 \times \mathbf{1}_1, \\ \mathbf{p}_5 &= -2 \times \mathbf{1}_2, \quad \mathbf{p}_6 = -\mathbf{1}_3, \\ \mathbf{y}_1 &= -3 \times \mathbf{1}_1, \quad \mathbf{y}_2 = -2 \times \mathbf{1}_2, \quad \mathbf{y}_3 = -\mathbf{1}_3, \quad \mathbf{y}_4 = 3 \times \mathbf{1}_1, \\ \mathbf{y}_5 &= 2 \times \mathbf{1}_2, \quad \mathbf{y}_6 = \mathbf{1}_3, \end{aligned}$$

where $\mathbf{1}_i$ is the i th column of the 3×3 identity matrix. The points in the first point-cloud are the centers of the faces of a rectangular prism and each point is associated with a point in the second point-cloud on the opposite face of another prism (that happens to be in the same location as the first)⁷.

Using these points, we have the following:

$$\mathbf{p} = \mathbf{0}, \quad \mathbf{y} = \mathbf{0}, \quad \mathbf{W} = \frac{1}{6} \text{diag}(-18, -8, -2), \quad (8.64)$$

which means the centroids are already on top of one another so we only need to rotate to align the point-clouds.

Using the ‘simplified approach’, we have

$$\mathbf{C} = (\mathbf{W}\mathbf{W}^T)^{-\frac{1}{2}} \mathbf{W} = \text{diag}(-1, -1, -1). \quad (8.65)$$

Unfortunately, we can easily see that $\det \mathbf{C} = -1$ and so $\mathbf{C} \notin SO(3)$, which indicates this approach has failed.

For the more rigorous approach, a singular-value decomposition of \mathbf{W} is

$$\begin{aligned} \mathbf{W} &= \mathbf{UDV}^T, & \mathbf{U} &= \text{diag}(1, 1, 1), & \mathbf{D} &= \frac{1}{6} \text{diag}(18, 8, 2), \\ & & \mathbf{V} &= \text{diag}(-1, -1, -1). \end{aligned} \quad (8.66)$$

We have $\det \mathbf{W} = -4/3 < 0$ and see that there is a unique minimum singular value, so we need to use the solution from *subcase (i-b)*. The minimal solution is therefore of the form $\mathbf{C} = \mathbf{USV}^T$ with $\mathbf{S} = \text{diag}(1, 1, -1)$. Plugging this in, we find

$$\mathbf{C} = \text{diag}(-1, -1, 1), \quad (8.67)$$

so that $\det \mathbf{C} = 1$. This is a rotation about the $\mathbf{1}_3$ axis through an angle π , which brings the error on four of the points to zero and leaves two of the points with non-zero error. This brings the objective function down to its minimum of $J = 4$.

Testing for Local Minima

In the previous section, we searched for global minima to the point-alignment problem and found there could be one or infinitely many. We did not, however, identify whether it was possible for local minima to exist, which we study now. Looking back to (8.30), this is the condition for a critical point in our optimization problem and therefore any solution that satisfies this criterion could be a minimum, a maximum, or a saddle point of the objective function, J .

If we have a solution, $\mathbf{C} \in SO(3)$, that satisfies (8.30), and we want

⁷ As a physical interpretation, imagine joining each of the six point pairs by rubber bands. Finding the \mathbf{C} that minimizes our cost metric is the same as finding the rotation that minimizes the amount of elastic energy stored in the rubber bands.

to characterize it, we can try perturbing the solution slightly and see whether the objective function goes up or down (or both). Consider a perturbation of the form

$$\mathbf{C}' = \exp(\phi^\wedge) \mathbf{C}, \quad (8.68)$$

where $\phi \in \mathbb{R}^3$ is a perturbation in an arbitrary direction, but constrained to keep $\mathbf{C}' \in SO(3)$. The change in the objective function δJ by applying the perturbation is

$$\delta J = J(\mathbf{C}') - J(\mathbf{C}) = -\text{tr}(\mathbf{C}' \mathbf{W}^T) + \text{tr}(\mathbf{C} \mathbf{W}^T) = -\text{tr}((\mathbf{C}' - \mathbf{C}) \mathbf{W}^T), \quad (8.69)$$

where we have neglected the Lagrange multiplier terms by assuming the perturbation keeps $\mathbf{C}' \in SO(3)$.

Now, approximating the perturbation out to second order, since this will tell us about the nature of the critical points, we have

$$\begin{aligned} \delta J &\approx -\text{tr}\left(\left(\left(\mathbf{1} + \phi^\wedge + \frac{1}{2}\phi^\wedge\phi^\wedge\right)\mathbf{C} - \mathbf{C}\right)\mathbf{W}^T\right) \\ &= -\text{tr}(\phi^\wedge \mathbf{C} \mathbf{W}^T) - \frac{1}{2}\text{tr}(\phi^\wedge \phi^\wedge \mathbf{C} \mathbf{W}^T). \end{aligned} \quad (8.70)$$

Then, plugging in the conditions for a critical point from (8.30), we have

$$\delta J = -\text{tr}(\phi^\wedge \mathbf{U} \mathbf{Y} \mathbf{D} \mathbf{S} \mathbf{Y}^T \mathbf{U}^T) - \frac{1}{2}\text{tr}(\phi^\wedge \phi^\wedge \mathbf{U} \mathbf{Y} \mathbf{D} \mathbf{S} \mathbf{Y}^T \mathbf{U}^T). \quad (8.71)$$

It turns out that the first term is zero (because it is a critical point), which we can see from

$$\begin{aligned} \text{tr}(\phi^\wedge \mathbf{U} \mathbf{Y} \mathbf{D} \mathbf{S} \mathbf{Y}^T \mathbf{U}^T) &= \text{tr}(\mathbf{Y}^T \mathbf{U}^T \phi^\wedge \mathbf{U} \mathbf{Y} \mathbf{D} \mathbf{S}) \\ &= \text{tr}((\mathbf{Y}^T \mathbf{U}^T \phi)^\wedge \mathbf{D} \mathbf{S}) = \text{tr}(\varphi^\wedge \mathbf{D} \mathbf{S}) = 0, \end{aligned} \quad (8.72)$$

where

$$\varphi = \begin{bmatrix} \varphi_1 \\ \varphi_2 \\ \varphi_3 \end{bmatrix} = \mathbf{Y}^T \mathbf{U}^T \phi, \quad (8.73)$$

and owing to the properties of a skew-symmetric matrix (zeros on the diagonal). For the second term, we use the identity $\mathbf{u}^\wedge \mathbf{u}^\wedge = -\mathbf{u}^T \mathbf{u} \mathbf{1} + \mathbf{u} \mathbf{u}^T$ to write

$$\begin{aligned} \delta J &= -\frac{1}{2}\text{tr}(\phi^\wedge \phi^\wedge \mathbf{U} \mathbf{Y} \mathbf{D} \mathbf{S} \mathbf{Y}^T \mathbf{U}^T) \\ &= -\frac{1}{2}\text{tr}\left(\mathbf{Y}^T \mathbf{U}^T \left(-\phi^T \phi \mathbf{1} + \phi \phi^T\right) \mathbf{U} \mathbf{Y} \mathbf{D} \mathbf{S}\right) \\ &= -\frac{1}{2}\text{tr}((- \varphi^2 \mathbf{1} + \varphi \varphi^T) \mathbf{D} \mathbf{S}), \end{aligned} \quad (8.74)$$

where $\varphi^2 = \boldsymbol{\varphi}^T \boldsymbol{\varphi} = \varphi_1^2 + \varphi_2^2 + \varphi_3^2$.

Manipulating a little further, we have

$$\begin{aligned}\delta J &= \frac{1}{2} \varphi^2 \text{tr}(\mathbf{DS}) - \frac{1}{2} \boldsymbol{\varphi}^T \mathbf{DS} \boldsymbol{\varphi} \\ &= \frac{1}{2} (\varphi_1^2(d_2 s_2 + d_3 s_3) + \varphi_2^2(d_1 s_1 + d_3 s_3) + \varphi_3^2(d_1 s_1 + d_2 s_2)),\end{aligned}\quad (8.75)$$

the sign of which depends entirely on the nature of \mathbf{DS} .

We can verify the ability of this expression to test for a minimum using the unique global minima identified in the previous section. For *subcase (i-a)*, where $d_1 \geq d_2 \geq d_3$ and $s_1 = s_2 = s_3$, we have

$$\delta J = \frac{1}{2} (\varphi_1^2(d_2 + d_3) + \varphi_2^2(d_1 + d_3) + \varphi_3^2(d_1 + d_2)) > 0 \quad (8.76)$$

for all $\boldsymbol{\varphi} \neq \mathbf{0}$, confirming a minimum. For *subcase (i-b)* where $d_1 \geq d_2 > d_3 > 0$ and $s_1 = s_2 = 1, s_3 = -1$, we have

$$\delta J = \frac{1}{2} (\underbrace{\varphi_1^2(d_2 - d_3)}_{>0} + \underbrace{\varphi_2^2(d_1 - d_3)}_{>0} + \varphi_3^2(d_1 + d_2)) > 0, \quad (8.77)$$

for all $\boldsymbol{\varphi} \neq \mathbf{0}$, again confirming a minimum. Finally, for *subcase (ii-a)* where $d_1 \geq d_2 > d_3 = 0$ and $s_1 = s_2 = 1, s_3 = \pm 1$, we have

$$\delta J = \frac{1}{2} (\varphi_1^2 d_2 + \varphi_2^2 d_1 + \varphi_3^2(d_1 + d_2)) > 0, \quad (8.78)$$

for all $\boldsymbol{\varphi} \neq \mathbf{0}$, once again confirming a minimum.

The more interesting question is whether there are any other local minima to worry about or not. This will become important when we use iterative methods to optimize rotation and pose variables. For example, let us consider *subcase (i-a)* a little further in the case that $d_1 > d_2 > d_3 > 0$. There are some other ways to satisfy (8.30) and generate a critical point. For example, we could pick $s_1 = s_2 = -1$ and $s_3 = 1$ so that $\det \mathbf{S} = 1$. In this case we have

$$\delta J = \frac{1}{2} (\underbrace{\varphi_1^2(d_3 - d_2)}_{<0} + \underbrace{\varphi_2^2(d_3 - d_1)}_{<0} + \underbrace{\varphi_3^2(-d_1 - d_2)}_{<0}) < 0, \quad (8.79)$$

which corresponds to a maximum since any $\boldsymbol{\varphi} \neq \mathbf{0}$ will decrease the objective function. The other two cases, $\mathbf{S} = \text{diag}(-1, 1, -1)$ and $\mathbf{S} = \text{diag}(1, -1, -1)$, turn out to be saddle points since depending on the direction of the perturbation, the objective function can go up or down. Since there are no other critical points, we can conclude there are no local minima other than the global one.

Similarly for *subcase (i-b)*, we need $\det \mathbf{S} = -1$ and can show that $\mathbf{S} = \text{diag}(-1, -1, -1)$ is a maximum and that $\mathbf{S} = \text{diag}(-1, 1, 1)$ and $\mathbf{S} = \text{diag}(1, -1, 1)$ are saddle points. Again, since there are no other

critical points, we can conclude there are no local minima other than the global one.

Also, for *subcase (ii-a)* we in general have

$$\delta J = \frac{1}{2} (\varphi_1^2 d_2 s_2 + \varphi_2^2 d_1 s_1 + \varphi_3^2 (d_1 s_1 + d_2 s_2)), \quad (8.80)$$

and so the only way to create a local minimum is to pick $s_1 = s_2 = 1$, which is the global minimum we have discussed earlier. Thus, again there are no additional local minima.

Iterative Approach

We can also consider using an iterative approach to solve for the optimal rotation matrix, \mathbf{C} . We will use our $SO(3)$ -sensitive scheme to do this. Importantly, the optimization we carry out is unconstrained, thereby avoiding the difficulties of the previous two approaches⁸. Technically, the result is not valid globally, only locally, as we require an initial guess that is refined from one iteration to the next; typically only a few iterations are needed. However, based on our discussion of local minima in the last section, we know that in all the important situations where there is a unique global minimum, there are no additional local minima to worry about.

Starting from the cost function where the translation has been eliminated,

$$J(\mathbf{C}) = \frac{1}{2} \sum_{j=1}^M w_j ((\mathbf{y}_j - \mathbf{y}) - \mathbf{C}(\mathbf{p}_j - \mathbf{p}))^T ((\mathbf{y}_j - \mathbf{y}) - \mathbf{C}(\mathbf{p}_j - \mathbf{p})), \quad (8.81)$$

we can insert the $SO(3)$ -sensitive perturbation,

$$\mathbf{C} = \exp(\psi^\wedge) \mathbf{C}_{\text{op}} \approx (1 + \psi^\wedge) \mathbf{C}_{\text{op}}, \quad (8.82)$$

where \mathbf{C}_{op} is the current guess and ψ is the perturbation; we will seek an optimal value to update the guess (and then iterate). Inserting the approximate perturbation scheme into the cost function turns it into a quadratic in ψ for which the minimizing value, ψ^* , is given by the solution to

$$\begin{aligned} & \underbrace{\mathbf{C}_{\text{op}} \left(-\frac{1}{w} \sum_{j=1}^M w_j (\mathbf{p}_j - \mathbf{p})^\wedge (\mathbf{p}_j - \mathbf{p})^\wedge \right) \mathbf{C}_{\text{op}}^T \psi^*}_{\text{constant}} \\ &= -\frac{1}{w} \sum_{j=1}^M w_j (\mathbf{y}_j - \mathbf{y})^\wedge \mathbf{C}_{\text{op}} (\mathbf{p}_j - \mathbf{p}). \end{aligned} \quad (8.83)$$

⁸ The iterative approach does not require solving either an eigenproblem nor carrying out a singular-value decomposition.

At first glance, the right-hand side appears to require recalculation using the individual points at each iteration. Fortunately, we can manipulate it into a more useful form. The right-hand side is a 3×1 column, and its i th row is given by

$$\begin{aligned} \mathbf{1}_i^T & \left(-\frac{1}{w} \sum_{j=1}^M w_j (\mathbf{y}_j - \mathbf{y})^\wedge \mathbf{C}_{\text{op}} (\mathbf{p}_j - \mathbf{p}) \right) \\ & = \frac{1}{w} \sum_{j=1}^M w_j (\mathbf{y}_j - \mathbf{y})^T \mathbf{1}_i^\wedge \mathbf{C}_{\text{op}} (\mathbf{p}_j - \mathbf{p}) \\ & = \frac{1}{w} \sum_{j=1}^M w_j \text{tr} (\mathbf{1}_i^\wedge \mathbf{C}_{\text{op}} (\mathbf{p}_j - \mathbf{p}) (\mathbf{y}_j - \mathbf{y})^T) \\ & = \text{tr} (\mathbf{1}_i^\wedge \mathbf{C}_{\text{op}} \mathbf{W}^T), \quad (8.84) \end{aligned}$$

where

$$\mathbf{W} = \frac{1}{w} \sum_{j=1}^M w_j (\mathbf{y}_j - \mathbf{y}) (\mathbf{p}_j - \mathbf{p})^T, \quad (8.85)$$

which we already saw in the non-iterative solution. Letting

$$\mathbf{I} = -\frac{1}{w} \sum_{j=1}^M w_j (\mathbf{p}_j - \mathbf{p})^\wedge (\mathbf{p}_j - \mathbf{p})^\wedge, \quad (8.86a)$$

$$\mathbf{b} = [\text{tr} (\mathbf{1}_i^\wedge \mathbf{C}_{\text{op}} \mathbf{W}^T)]_i, \quad (8.86b)$$

the optimal update can be written in closed form as

$$\boldsymbol{\psi}^* = \mathbf{C}_{\text{op}} \mathbf{I}^{-1} \mathbf{C}_{\text{op}}^T \mathbf{b}. \quad (8.87)$$

We apply this to the initial guess,

$$\mathbf{C}_{\text{op}} \leftarrow \exp(\boldsymbol{\psi}^{*\wedge}) \mathbf{C}_{\text{op}}, \quad (8.88)$$

and iterate to convergence, taking $\hat{\mathbf{C}}_{v_k i} = \mathbf{C}_{\text{op}}$ at the final iteration as our rotation estimate. After convergence, the translation is given as in the non-iterative scheme:

$$\hat{\mathbf{r}}_i^{v_k i} = \mathbf{p} - \hat{\mathbf{C}}_{v_k i}^T \mathbf{y}. \quad (8.89)$$

Notably, both \mathbf{I} and \mathbf{W} can be computed in advance, and therefore we do not require the original points during execution of the iterative scheme.

Three Non-collinear Points Required

Clearly, to solve uniquely for $\boldsymbol{\psi}^*$ above, we need $\det \mathbf{I} \neq 0$. A sufficient condition is to have \mathbf{I} positive-definite, which implies that for any $\mathbf{x} \neq \mathbf{0}$, we must have

$$\mathbf{x}^T \mathbf{I} \mathbf{x} > 0. \quad (8.90)$$

We then notice that

$$\begin{aligned}\mathbf{x}^T \mathbf{I} \mathbf{x} &= \mathbf{x}^T \left(-\frac{1}{w} \sum_{j=1}^M w_j (\mathbf{p}_j - \mathbf{p})^\wedge (\mathbf{p}_j - \mathbf{p})^\wedge \right) \mathbf{x} \\ &= \frac{1}{w} \sum_{j=1}^M w_j \underbrace{((\mathbf{p}_j - \mathbf{p})^\wedge \mathbf{x})^T ((\mathbf{p}_j - \mathbf{p})^\wedge \mathbf{x})}_{\geq 0} \geq 0.\end{aligned}\quad (8.91)$$

Since each term in the sum is non-negative, the total must be non-negative. The only way to have the total be zero is if *every* term in the sum is also zero, or

$$(\forall j) \ (\mathbf{p}_j - \mathbf{p})^\wedge \mathbf{x} = \mathbf{0}. \quad (8.92)$$

In other words, we must have $\mathbf{x} = \mathbf{0}$ (not true by assumption), $\mathbf{p}_j = \mathbf{p}$, or \mathbf{x} parallel to $\mathbf{p}_j - \mathbf{p}$. The last two conditions are never true as long as there are at least three points and they are not collinear.

Note that having three non-collinear points only provides a sufficient condition for a unique solution for ψ^* at each iteration, and does not tell us about the number of possible global solutions to minimize our objective function in general. This was discussed at length in the previous sections, where we learned there could be one or infinitely many global solutions. Moreover, if there is a unique global minimum, there are no local minima to worry about.

8.1.4 Transformation Matrix Solution

Finally, for completeness, we also can provide an iterative approach to solving for the pose change using transformation matrices and their relationship to the exponential map⁹. As in the previous two sections, we will use some simplified notation to avoid repeating sub- and super-scripts:

$$\begin{aligned}\mathbf{y}_j &= \begin{bmatrix} \mathbf{y}_j \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{r}_{v_k}^{p_j v_k} \\ 1 \end{bmatrix}, \quad \mathbf{p}_j = \begin{bmatrix} \mathbf{p}_j \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{r}_i^{p_j i} \\ 1 \end{bmatrix}, \\ \mathbf{T} &= \mathbf{T}_{v_k i} = \begin{bmatrix} \mathbf{C}_{v_k i} & -\mathbf{C}_{v_k i} \mathbf{r}_i^{v_k i} \\ \mathbf{0}^T & 1 \end{bmatrix}.\end{aligned}\quad (8.93)$$

We have used a different font for the homogeneous representations of the points; we will be making connections back to the previous section on rotation matrices so we also keep the non-homogeneous point representations around for convenience.

We define our error term for each point as

$$\mathbf{e}_j = \mathbf{y}_j - \mathbf{T} \mathbf{p}_j, \quad (8.94)$$

⁹ We will use the optimization approach outlined in Section 7.1.9.

and our objective function as

$$J(\mathbf{T}) = \frac{1}{2} \sum_{j=1}^M w_j \mathbf{e}_j^T \mathbf{e}_j = \frac{1}{2} \sum_{j=1}^M w_j (\mathbf{y}_j - \mathbf{T}\mathbf{p}_j)^T (\mathbf{y}_j - \mathbf{T}\mathbf{p}_j), \quad (8.95)$$

where $w_j > 0$ are the usual scalar weights. We seek to minimize J with respect to $\mathbf{T} \in SE(3)$. Notably, this objective function is equivalent to the ones for the unit-quaternion and rotation-matrix parameterizations, so the minima should be the same.

To do this, we use our $SE(3)$ -sensitive perturbation scheme,

$$\mathbf{T} = \exp(\boldsymbol{\epsilon}^\wedge) \mathbf{T}_{\text{op}} \approx (\mathbf{1} + \boldsymbol{\epsilon}^\wedge) \mathbf{T}_{\text{op}}, \quad (8.96)$$

where \mathbf{T}_{op} is some initial guess (i.e., operating point of our linearization) and $\boldsymbol{\epsilon}$ is a small perturbation to that guess. Inserting this into the objective function, we then have

$$J(\mathbf{T}) \approx \frac{1}{2} \sum_{j=1}^M w_j ((\mathbf{y}_j - \mathbf{z}_j) - \mathbf{z}_j^\odot \boldsymbol{\epsilon})^T ((\mathbf{y}_j - \mathbf{z}_j) - \mathbf{z}_j^\odot \boldsymbol{\epsilon}), \quad (8.97)$$

where $\mathbf{z}_j = \mathbf{T}_{\text{op}} \mathbf{p}_j$ and we have used that

$$\boldsymbol{\epsilon}^\wedge \mathbf{z}_j = \mathbf{z}_j^\odot \boldsymbol{\epsilon}, \quad (8.98)$$

which was explained in Section 7.1.8.

Our objective function is now exactly quadratic in $\boldsymbol{\epsilon}$, and therefore we can carry out a simple, *unconstrained* optimization for $\boldsymbol{\epsilon}$. Taking the derivative, we find

$$\frac{\partial J}{\partial \boldsymbol{\epsilon}^T} = - \sum_{j=1}^M w_j \mathbf{z}_j^{\odot T} ((\mathbf{y}_j - \mathbf{z}_j) - \mathbf{z}_j^\odot \boldsymbol{\epsilon}). \quad (8.99)$$

Setting this to zero, we have the following system of equations for the optimal $\boldsymbol{\epsilon}^*$:

$$\left(\frac{1}{w} \sum_{j=1}^M w_j \mathbf{z}_j^{\odot T} \mathbf{z}_j^\odot \right) \boldsymbol{\epsilon}^* = \frac{1}{w} \sum_{j=1}^M w_j \mathbf{z}_j^{\odot T} (\mathbf{y}_j - \mathbf{z}_j). \quad (8.100)$$

While we could use this to compute the optimal update, both the left- and right-hand sides require construction from the original points at each iteration. As in the previous section on the iterative solution using rotation matrices, it turns out we can manipulate both sides into forms that do not require the original points.

Looking to the left-hand side first, we can show that

$$\frac{1}{w} \sum_{j=1}^M w_j \mathbf{z}_j^{\odot T} \mathbf{z}_j^\odot = \underbrace{\mathcal{T}_{\text{op}}^{-T}}_{>0} \underbrace{\left(\frac{1}{w} \sum_{j=1}^M w_j \mathbf{p}_j^{\odot T} \mathbf{p}_j^\odot \right)}_{\mathcal{M}} \underbrace{\mathcal{T}_{\text{op}}^{-1}}_{>0}, \quad (8.101)$$

where

$$\begin{aligned}\mathcal{T}_{\text{op}} &= \text{Ad}(\mathbf{T}_{\text{op}}), \quad \mathbf{\mathcal{M}} = \begin{bmatrix} \mathbf{1} & \mathbf{0} \\ \mathbf{p}^\wedge & \mathbf{1} \end{bmatrix} \begin{bmatrix} \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{1} & -\mathbf{p}^\wedge \\ \mathbf{0} & \mathbf{1} \end{bmatrix}, \\ w &= \sum_{j=1}^M w_j, \quad \mathbf{p} = \frac{1}{w} \sum_{j=1}^M w_j \mathbf{p}_j, \quad \mathbf{I} = -\frac{1}{w} \sum_{j=1}^M w_j (\mathbf{p}_j - \mathbf{p})^\wedge (\mathbf{p}_j - \mathbf{p})^\wedge.\end{aligned}\tag{8.102}$$

The 6×6 matrix, $\mathbf{\mathcal{M}}$, has the form of a *generalized mass matrix* (Murray et al., 1994) with the weights as surrogates for masses. Notably, it is only a function of the points in the stationary frame and is therefore a constant.

Looking to the right-hand side, we can also show that

$$\mathbf{a} = \frac{1}{w} \sum_{j=1}^M w_j \mathbf{z}_j^{\odot T} (\mathbf{y}_j - \mathbf{z}_j) = \begin{bmatrix} \mathbf{y} - \mathbf{C}_{\text{op}}(\mathbf{p} - \mathbf{r}_{\text{op}}) \\ \mathbf{b} - \mathbf{y}^\wedge \mathbf{C}_{\text{op}}(\mathbf{p} - \mathbf{r}_{\text{op}}) \end{bmatrix},\tag{8.103}$$

where

$$\mathbf{b} = [\text{tr}(\mathbf{1}_i^\wedge \mathbf{C}_{\text{op}} \mathbf{W}^T)]_i, \quad \mathbf{T}_{\text{op}} = \begin{bmatrix} \mathbf{C}_{\text{op}} & -\mathbf{C}_{\text{op}} \mathbf{r}_{\text{op}} \\ \mathbf{0}^T & 1 \end{bmatrix},\tag{8.104}$$

$$\mathbf{W} = \frac{1}{w} \sum_{j=1}^M w_j (\mathbf{y}_j - \mathbf{y})(\mathbf{p}_j - \mathbf{p})^T, \quad \mathbf{y} = \frac{1}{w} \sum_{j=1}^M w_j \mathbf{y}_j.\tag{8.105}$$

Both \mathbf{W} and \mathbf{y} we have seen before and can be computed in advance from the points and then used at each iteration of the scheme.

Once again, we can write the solution for the optimal update down in closed form:

$$\boldsymbol{\epsilon}^* = \mathcal{T}_{\text{op}} \mathbf{\mathcal{M}}^{-1} \mathcal{T}_{\text{op}}^T \mathbf{a}.\tag{8.106}$$

Once computed, we simply update our operating point,

$$\mathbf{T}_{\text{op}} \leftarrow \exp(\boldsymbol{\epsilon}^*) \mathbf{T}_{\text{op}},\tag{8.107}$$

and iterate the procedure to convergence. The estimated transformation is then $\hat{\mathbf{T}}_{v_k i} = \mathbf{T}_{\text{op}}$ at the final iteration. Alternatively, $\hat{\mathbf{T}}_{iv_k} = \hat{\mathbf{T}}_{v_k i}^{-1}$ may be the output of interest.

Note, applying the optimal perturbation through the exponential map ensures that \mathbf{T}_{op} remains in $SE(3)$ at each iteration. Also, looking back to Section 4.3.1, we can see that our iterative optimization of \mathbf{T} is exactly in the form of a Gauss-Newton style estimator, but adapted to work with $SE(3)$.

Three Non-collinear Points Required

It is interesting to consider when (8.100) has a unique solution. It immediately follows from (8.101) that

$$\det \mathbf{\mathcal{M}} = \det \mathbf{I}.\tag{8.108}$$

Therefore, to uniquely solve for ϵ^* above, we need $\det \mathbf{I} \neq 0$. A sufficient condition is to have \mathbf{I} positive-definite, which (we saw in the previous section on rotation matrices) is true as long as there are at least three points and they are not collinear.

8.2 Point-Cloud Tracking

In this section, we study a problem very much related to point-cloud alignment, namely, point-cloud tracking. In the alignment problem, we simply wanted to align two point-clouds to determine the vehicle's pose at a single time. In the tracking problem, we want to estimate the pose of an object over time through a combination of measurements and a prior (with inputs). Accordingly, we will set up motion and observation models and then show how we can use these in both recursive (i.e., EKF) and batch (i.e., Gauss-Newton) solutions.

8.2.1 Problem Setup

We will continue to use the situation depicted in Figure 8.1. The state of the vehicle comprises

$$\begin{aligned}\mathbf{r}_i^{v_k i} &: \text{translation vector from } I \text{ to } V_k, \text{ expressed in } \underline{\mathcal{F}}_i \\ \mathbf{C}_{v_k i} &: \text{rotation matrix from } \underline{\mathcal{F}}_i \text{ to } \underline{\mathcal{F}}_{v_k}\end{aligned}$$

or alternatively,

$$\mathbf{T}_k = \mathbf{T}_{v_k i} = \begin{bmatrix} \mathbf{C}_{v_k i} & -\mathbf{C}_{v_k i} \mathbf{r}_i^{v_k i} \\ \mathbf{0}^T & 1 \end{bmatrix}, \quad (8.109)$$

as a single transformation matrix. We use the shorthand

$$\mathbf{x} = \{\mathbf{T}_0, \mathbf{T}_1, \dots, \mathbf{T}_K\}, \quad (8.110)$$

for the entire trajectory of poses. Our motion prior/inputs and measurements for this problem are as follows:

(i) Motion Prior/Inputs:

- We might assume the known inputs are the initial pose (with uncertainty),

$$\check{\mathbf{T}}_0, \quad (8.111)$$

as well as the translational velocity, $\boldsymbol{\nu}_{v_k}^{iv_k}$, and angular velocity of the vehicle, $\boldsymbol{\omega}_{v_k}^{iv_k}$, which we note are expressed in the vehicle frame. We combine these as

$$\boldsymbol{\varpi}_k = \begin{bmatrix} \boldsymbol{\nu}_{v_k}^{iv_k} \\ \boldsymbol{\omega}_{v_k}^{iv_k} \end{bmatrix}, \quad k = 1 \dots K, \quad (8.112)$$

at a number of discrete times (we will assume the inputs

are piecewise-constant in time). Together, the inputs can be written using the shorthand,

$$\mathbf{v} = \{\check{\mathbf{T}}_0, \boldsymbol{\varpi}_1, \boldsymbol{\varpi}_2, \dots, \boldsymbol{\varpi}_K\}. \quad (8.113)$$

(ii) Measurements:

- We assume we are capable of measuring the position of a particular stationary point, P_j , in the vehicle frame, $\mathbf{r}_{v_k}^{p_j v_k}$. We assume the position of the point is known in the stationary frame, $\mathbf{r}_i^{p_j i}$. Note that there could also be measurements of multiple points, hence the subscript j . We will write

$$\mathbf{y}_{jk} = \mathbf{r}_{v_k}^{p_j v_k}, \quad (8.114)$$

for the observation of point P_j at discrete time k . Together, the measurements can be written using the shorthand,

$$\mathbf{y} = \{\mathbf{y}_{11}, \dots, \mathbf{y}_{M1}, \dots, \mathbf{y}_{1K}, \dots, \mathbf{y}_{MK}\}. \quad (8.115)$$

This pose estimation problem is fairly generic and could be used to describe a variety of situations.

8.2.2 Motion Priors

We will derive a general discrete-time, kinematic motion prior that can be used within a number of different estimation algorithms. We will start in continuous time and then move to discrete time.

Continuous Time

We will start with the $SE(3)$ kinematics¹⁰,

$$\dot{\mathbf{T}} = \boldsymbol{\varpi}^\wedge \mathbf{T}, \quad (8.116)$$

where the quantities involved are perturbed by process noise according to

$$\mathbf{T} = \exp(\delta\xi^\wedge) \bar{\mathbf{T}}, \quad (8.117a)$$

$$\boldsymbol{\varpi} = \bar{\boldsymbol{\varpi}} + \delta\boldsymbol{\varpi}. \quad (8.117b)$$

We can separate these into nominal and perturbation kinematics as in (7.253):

$$\text{nominal kinematics: } \dot{\bar{\mathbf{T}}} = \bar{\boldsymbol{\varpi}}^\wedge \bar{\mathbf{T}}, \quad (8.118a)$$

$$\text{perturbation kinematics: } \delta\dot{\xi} = \bar{\boldsymbol{\varpi}}^\wedge \delta\xi + \delta\boldsymbol{\varpi}, \quad (8.118b)$$

where we will think of $\delta\boldsymbol{\varpi}(t)$ as process noise that corrupts the nominal kinematics. Thus, integrating the perturbed kinematic equation allows

¹⁰ To be clear, $\mathbf{T} = \mathbf{T}_{v_k i}$ in this equation and $\boldsymbol{\varpi}$ is the generalized velocity expressed in $\underline{\mathcal{F}}_{v_k}$.

us to track uncertainty in the pose of the system. While we could do this in continuous time, we will next move to discrete time to prepare to use this kinematic model in the EKF and batch, discrete-time MAP estimators.

Discrete Time

If we assume quantities remain constant between discrete times, then we can use the ideas from Section 7.2.2 to write

$$\text{nominal kinematics: } \bar{\mathbf{T}}_k = \underbrace{\exp(\Delta t_k \bar{\boldsymbol{\omega}}_k^\wedge)}_{\Xi_k} \bar{\mathbf{T}}_{k-1}, \quad (8.119a)$$

$$\text{perturbation kinematics: } \delta\boldsymbol{\xi}_k = \underbrace{\exp(\Delta t_k \bar{\boldsymbol{\omega}}_k^\wedge)}_{\text{Ad}(\Xi_k)} \delta\boldsymbol{\xi}_{k-1} + \mathbf{w}_k, \quad (8.119b)$$

with $\Delta t_k = t_k - t_{k-1}$ for the nominal and perturbation kinematics in discrete time. The process noise is now $\mathbf{w}_k = \mathcal{N}(\mathbf{0}, \mathbf{Q}_k)$.

8.2.3 Measurement Model

We next develop a measurement model and then linearize it.

Nonlinear

Our 3×1 measurement model can be compactly written as

$$\mathbf{y}_{jk} = \mathbf{D}^T \mathbf{T}_k \mathbf{p}_j + \mathbf{n}_{jk}, \quad (8.120)$$

where the position of the known points on the moving vehicle are expressed in 4×1 homogeneous coordinates (bottom row equal to 1),

$$\mathbf{p}_j = \begin{bmatrix} \mathbf{r}_i^{p_j i} \\ 1 \end{bmatrix}, \quad (8.121)$$

and

$$\mathbf{D}^T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad (8.122)$$

is a projection matrix used to ensure the measurements are indeed 3×1 by removing the 1 on the bottom row. We have also now included, $\mathbf{n}_{jk} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_{jk})$, which is Gaussian measurement noise.

Linearized

We linearize (8.120) in much the same way as the motion model through the use of perturbations:

$$\mathbf{T}_k = \exp(\delta\boldsymbol{\xi}_k^\wedge) \bar{\mathbf{T}}_k, \quad (8.123a)$$

$$\mathbf{y}_{jk} = \bar{\mathbf{y}}_{jk} + \delta\mathbf{y}_{jk}. \quad (8.123b)$$

Substituting these into the measurement model, we have

$$\bar{\mathbf{y}}_{jk} + \delta\mathbf{y}_{jk} = \mathbf{D}^T (\exp(\delta\xi_k^\wedge) \bar{\mathbf{T}}_k) \mathbf{p}_j + \mathbf{n}_{jk}. \quad (8.124)$$

Subtracting off the nominal solution (i.e., the operating point in our linearization),

$$\bar{\mathbf{y}}_{jk} = \mathbf{D}^T \bar{\mathbf{T}}_k \mathbf{p}_j, \quad (8.125)$$

we are left with

$$\delta\mathbf{y}_{jk} \approx \mathbf{D}^T (\bar{\mathbf{T}}_k \mathbf{p}_j)^\odot \delta\xi_k + \mathbf{n}_{jk}, \quad (8.126)$$

correct to first order. This perturbation measurement model relates small changes in the input to the measurement model to small changes in the output, in an $SE(3)$ -constraint-sensitive manner.

Nomenclature

To match the notation used in our derivations of our nonlinear estimators, we define the following symbols:

- $\hat{\mathbf{T}}_k$: 4×4 corrected estimate of pose at time k
- $\hat{\mathbf{P}}_k$: 6×6 covariance of corrected estimate at time k
(for both translation and rotation)
- $\check{\mathbf{T}}_k$: 4×4 predicted estimate of pose at time k
- $\check{\mathbf{P}}_k$: 6×6 covariance of predicted estimate at time k
(for both translation and rotation)
- $\check{\mathbf{T}}_0$: 4×4 prior input as pose at time 0
- ϖ_k : 6×1 prior input as generalized velocity at time k
- \mathbf{Q}_k : 6×6 covariance of process noise
(for both translation and rotation)
- \mathbf{y}_{jk} : 3×1 measurement of point j from vehicle at time k
- \mathbf{R}_{jk} : 3×3 covariance of measurement j at time k

We will use these in two different estimators, the EKF and batch discrete-time MAP estimation.

8.2.4 EKF Solution

In this section, we seek to estimate the pose of our vehicle using the classic EKF, but carefully applied to our situation involving rotations.

Prediction Step

Predicting the mean forward in time is not difficult in the case of the EKF; we simply pass our prior estimate and latest input through the

nominal kinematics model in (8.119):

$$\check{\mathbf{T}}_k = \underbrace{\exp(\Delta t_k \boldsymbol{\varpi}_k^\wedge)}_{\Xi_k} \hat{\mathbf{T}}_{k-1}. \quad (8.127)$$

To predict the covariance of the estimate,

$$\check{\mathbf{P}}_k = E[\delta \check{\xi}_k \delta \check{\xi}_k^T], \quad (8.128)$$

we require the perturbation kinematics model in (8.119),

$$\delta \check{\xi}_k = \underbrace{\exp(\Delta t_k \boldsymbol{\varpi}_k^\wedge)}_{\mathbf{F}_{k-1} = \text{Ad}(\Xi_k)} \delta \hat{\xi}_{k-1} + \mathbf{w}_k. \quad (8.129)$$

Thus, in this case, the coefficient matrix of the linearized motion model is

$$\mathbf{F}_{k-1} = \exp(\Delta t_k \boldsymbol{\varpi}_k^\wedge), \quad (8.130)$$

which depends only on the input and not the state due to our convenient choice of representing uncertainty via the exponential map. The covariance prediction proceeds in the usual EKF manner as

$$\check{\mathbf{P}}_k = \mathbf{F}_{k-1} \hat{\mathbf{P}}_{k-1} \mathbf{F}_{k-1}^T + \mathbf{Q}_k. \quad (8.131)$$

The corrective step is where we must pay particular attention to the pose variables.

Correction Step

Looking back to (8.126) for the perturbation measurement model,

$$\delta \mathbf{y}_{jk} = \underbrace{\mathbf{D}^T (\check{\mathbf{T}}_k \mathbf{p}_j)^\odot}_{\mathbf{G}_{jk}} \delta \check{\xi}_k + \mathbf{n}_{jk}, \quad (8.132)$$

we see that the coefficient matrix of the linearized measurement model is

$$\mathbf{G}_{jk} = \mathbf{D}^T (\check{\mathbf{T}}_k \mathbf{p}_j)^\odot, \quad (8.133)$$

which is evaluated at the predicted mean pose, $\check{\mathbf{T}}_k$.

To handle the case in which there are M observations of points on the vehicle, we can stack the quantities as follows:

$$\mathbf{y}_k = \begin{bmatrix} \mathbf{y}_{1k} \\ \vdots \\ \mathbf{y}_{Mk} \end{bmatrix}, \quad \mathbf{G}_k = \begin{bmatrix} \mathbf{G}_{1k} \\ \vdots \\ \mathbf{G}_{Mk} \end{bmatrix}, \quad \mathbf{R}_k = \text{diag}(\mathbf{R}_{1k}, \dots, \mathbf{R}_{Mk}). \quad (8.134)$$

The Kalman gain and covariance update equations are then unchanged from the generic case:

$$\mathbf{K}_k = \check{\mathbf{P}}_k \mathbf{G}_k^T (\mathbf{G}_k \check{\mathbf{P}}_k \mathbf{G}_k^T + \mathbf{R}_k)^{-1}, \quad (8.135a)$$

$$\hat{\mathbf{P}}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{G}_k) \check{\mathbf{P}}_k. \quad (8.135b)$$

Note that we must be careful to interpret the EKF corrective equations properly since

$$\hat{\mathbf{P}}_k = E \left[\delta \hat{\boldsymbol{\xi}}_k \delta \hat{\boldsymbol{\xi}}_k^T \right]. \quad (8.136)$$

In particular, for the mean update, we rearrange the equation as follows:

$$\boldsymbol{\epsilon}_k = \underbrace{\ln \left(\hat{\mathbf{T}}_k \check{\mathbf{T}}_k^{-1} \right)^\vee}_{\text{update}} = \mathbf{K}_k \underbrace{(\mathbf{y}_k - \check{\mathbf{y}}_k)}_{\text{innovation}}, \quad (8.137)$$

where $\boldsymbol{\epsilon}_k = \ln \left(\hat{\mathbf{T}}_k \check{\mathbf{T}}_k^{-1} \right)^\vee$ is the difference of the corrected and predicted means and $\check{\mathbf{y}}_k$ is the nonlinear, nominal measurement model evaluated at the predicted mean:

$$\check{\mathbf{y}}_k = \begin{bmatrix} \check{\mathbf{y}}_{1k} \\ \vdots \\ \check{\mathbf{y}}_{Mk} \end{bmatrix}, \quad \check{\mathbf{y}}_{jk} = \mathbf{D}^T \check{\mathbf{T}}_k \mathbf{p}_j, \quad (8.138)$$

where we have again accounted for the fact that there could be M observations of points on the vehicle. Once we have computed the mean correction, $\boldsymbol{\epsilon}_k$, we apply it according to

$$\check{\mathbf{T}}_k = \exp(\boldsymbol{\epsilon}_k^\wedge) \check{\mathbf{T}}_k, \quad (8.139)$$

which ensures the mean stays in $SE(3)$.

Summary

Putting the pieces from the last two sections together, we have our canonical five EKF equations for this system:

$$\text{predictor: } \check{\mathbf{P}}_k = \mathbf{F}_{k-1} \hat{\mathbf{P}}_{k-1} \mathbf{F}_{k-1}^T + \mathbf{Q}_k, \quad (8.140a)$$

$$\check{\mathbf{T}}_k = \Xi_k \hat{\mathbf{T}}_{k-1}, \quad (8.140b)$$

$$\text{Kalman gain: } \mathbf{K}_k = \check{\mathbf{P}}_k \mathbf{G}_k^T (\mathbf{G}_k \check{\mathbf{P}}_k \mathbf{G}_k^T + \mathbf{R}_k)^{-1}, \quad (8.140c)$$

$$\text{corrector: } \hat{\mathbf{P}}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{G}_k) \check{\mathbf{P}}_k, \quad (8.140d)$$

$$\hat{\mathbf{T}}_k = \exp((\mathbf{K}_k (\mathbf{y}_k - \check{\mathbf{y}}_k))^\wedge) \check{\mathbf{T}}_k. \quad (8.140e)$$

We have essentially modified the EKF so that all the mean calculations occur in $SE(3)$, the Lie group, and all of the covariance calculations occur in $\mathfrak{se}(3)$, the Lie algebra. As usual, we must initialize the filter at the first timestep using $\check{\mathbf{T}}_0$. Although we do not show it, we could easily turn this into an iterated EKF by relinearizing about the latest estimate and iterating over the correction step. Finally, the algorithm has $\hat{\mathbf{T}}_{v_k i} = \hat{\mathbf{T}}_k$ so we can compute $\hat{\mathbf{T}}_{iv_k} = \hat{\mathbf{T}}_k^{-1}$ if desired.

8.2.5 Batch Maximum a Posteriori Solution

In this section, we return to the discrete-time, batch estimation approach to see how this works on our pose tracking problem.

Error Terms and Objective Function

As usual for batch MAP problems, we begin by defining an error term for each of our inputs and measurements. For the inputs, $\check{\mathbf{T}}_0$ and $\boldsymbol{\varpi}_k$, we have

$$\mathbf{e}_{v,k}(\mathbf{x}) = \begin{cases} \ln (\check{\mathbf{T}}_0 \mathbf{T}_0^{-1})^\vee & k = 0 \\ \ln (\boldsymbol{\Xi}_k \mathbf{T}_{k-1} \mathbf{T}_k^{-1})^\vee & k = 1 \dots K \end{cases}, \quad (8.141)$$

where $\boldsymbol{\Xi}_k = \exp(\Delta t_k \boldsymbol{\varpi}_k^\wedge)$ and we have used the convenient shorthand, $\mathbf{x} = \{\mathbf{T}_0, \dots, \mathbf{T}_K\}$. For the measurements, \mathbf{y}_{jk} , we have

$$\mathbf{e}_{y,jk}(\mathbf{x}) = \mathbf{y}_{jk} - \mathbf{D}^T \mathbf{T}_k \mathbf{p}_j. \quad (8.142)$$

Next we examine the noise properties of these errors.

Taking the Bayesian point of view, we consider that the true pose variables are drawn from the prior (see Section 4.1.1) so that

$$\mathbf{T}_k = \exp(\delta \boldsymbol{\xi}_k^\wedge) \check{\mathbf{T}}_k, \quad (8.143)$$

where $\delta \boldsymbol{\xi}_k \sim \mathcal{N}(\mathbf{0}, \check{\mathbf{P}}_k)$.

For the first input error, we have

$$\mathbf{e}_{v,0}(\mathbf{x}) = \ln (\check{\mathbf{T}}_0 \mathbf{T}_0^{-1})^\vee = \ln (\check{\mathbf{T}}_0 \check{\mathbf{T}}_0^{-1} \exp(-\delta \boldsymbol{\xi}_0^\wedge))^\vee = -\delta \boldsymbol{\xi}_0, \quad (8.144)$$

so that

$$\mathbf{e}_{v,0}(\mathbf{x}) \sim \mathcal{N}(\mathbf{0}, \check{\mathbf{P}}_0). \quad (8.145)$$

For the later input errors, we have

$$\begin{aligned} \mathbf{e}_{v,k}(\mathbf{x}) &= \ln (\boldsymbol{\Xi}_k \mathbf{T}_{k-1} \mathbf{T}_k^{-1})^\vee \\ &= \ln (\boldsymbol{\Xi}_k \exp(\delta \boldsymbol{\xi}_{k-1}^\wedge) \check{\mathbf{T}}_{k-1} \check{\mathbf{T}}_k^{-1} \exp(-\delta \boldsymbol{\xi}_k^\wedge))^\vee \\ &= \ln \left(\underbrace{\boldsymbol{\Xi}_k \check{\mathbf{T}}_{k-1} \check{\mathbf{T}}_k^{-1}}_1 \exp \left((\text{Ad}(\boldsymbol{\Xi}_k) \delta \boldsymbol{\xi}_{k-1})^\wedge \right) \exp(-\delta \boldsymbol{\xi}_k^\wedge) \right)^\vee \\ &\approx \text{Ad}(\boldsymbol{\Xi}_k) \delta \boldsymbol{\xi}_{k-1} - \delta \boldsymbol{\xi}_k \\ &= -\mathbf{w}_k, \end{aligned} \quad (8.146)$$

so that

$$\mathbf{e}_{v,k}(\mathbf{x}) \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_k). \quad (8.147)$$

For the measurement model, we consider that the measurements are generated by evaluating the noise-free versions (based on the true pose variables) and then corrupted by noise so that

$$\mathbf{e}_{y,jk}(\mathbf{x}) = \mathbf{y}_{jk} - \mathbf{D}^T \mathbf{T}_k \mathbf{p}_j = \mathbf{n}_{jk}, \quad (8.148)$$

so that

$$\mathbf{e}_{y,jk}(\mathbf{x}) \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_{jk}). \quad (8.149)$$

These noise properties allow us to next construct the objective function that we want to minimize in our batch MAP problem:

$$J_{v,k}(\mathbf{x}) = \begin{cases} \frac{1}{2}\mathbf{e}_{v,0}(\mathbf{x})^T \check{\mathbf{P}}_0^{-1} \mathbf{e}_{v,0}(\mathbf{x}) & k=0 \\ \frac{1}{2}\mathbf{e}_{v,k}(\mathbf{x})^T \mathbf{Q}_k^{-1} \mathbf{e}_{v,k}(\mathbf{x}) & k=1\dots K \end{cases}, \quad (8.150a)$$

$$J_{y,k}(\mathbf{x}) = \frac{1}{2}\mathbf{e}_{y,k}(\mathbf{x})^T \mathbf{R}_k^{-1} \mathbf{e}_{y,k}(\mathbf{x}), \quad (8.150b)$$

where we have stacked the M point quantities together according to

$$\mathbf{e}_{y,k}(\mathbf{x}) = \begin{bmatrix} \mathbf{e}_{y,1k}(\mathbf{x}) \\ \vdots \\ \mathbf{e}_{y,Mk}(\mathbf{x}) \end{bmatrix}, \quad \mathbf{R}_k = \text{diag}(\mathbf{R}_{1k}, \dots, \mathbf{R}_{Mk}). \quad (8.151)$$

The overall objective function that we will seek to minimize is then

$$J(\mathbf{x}) = \sum_{k=0}^K (J_{v,k}(\mathbf{x}) + J_{y,k}(\mathbf{x})). \quad (8.152)$$

the next section will look at linearizing our error terms in order to carry out Gauss-Newton optimization.

Linearized Error Terms

It is fairly straightforward to linearize our error terms (in order to carry out Gauss-Newton optimization) just as we earlier linearized our motion and observation models. We will linearize about an operating point for each pose, $\mathbf{T}_{\text{op},k}$, which we can think of as our current trajectory guess that will be iteratively improved. Thus, we will take

$$\mathbf{T}_k = \exp(\boldsymbol{\epsilon}_k^\wedge) \mathbf{T}_{\text{op},k}, \quad (8.153)$$

where $\boldsymbol{\epsilon}_k$ will be the perturbation to the current guess that we seek to optimize at each iteration. We will use the shorthand

$$\mathbf{x}_{\text{op}} = \{\mathbf{T}_{\text{op},1}, \mathbf{T}_{\text{op},2}, \dots, \mathbf{T}_{\text{op},K}\}, \quad (8.154)$$

for the operating point of the entire trajectory.

For the first input error, we have

$$\mathbf{e}_{v,0}(\mathbf{x}) = \ln(\check{\mathbf{T}}_0 \mathbf{T}_0^{-1})^\vee = \ln\left(\underbrace{\check{\mathbf{T}}_0 \mathbf{T}_{\text{op},0}^{-1}}_{\exp(\mathbf{e}_{v,0}(\mathbf{x}_{\text{op}})^\wedge)} \exp(-\boldsymbol{\epsilon}_0^\wedge)\right)^\vee \approx \mathbf{e}_{v,0}(\mathbf{x}_{\text{op}}) - \boldsymbol{\epsilon}_0, \quad (8.155)$$

where $\mathbf{e}_{v,0}(\mathbf{x}_{\text{op}}) = \ln(\check{\mathbf{T}}_0 \mathbf{T}_{\text{op},0}^{-1})^\vee$ is the error evaluated at the operating point. Note, we have used a very crude version of the BCH formula to arrive at the approximation on the right (i.e., only the first two terms),

but this approximation will get better as ϵ_0 goes to zero, which will happen as the Gauss-Newton algorithm converges¹¹.

For the input errors at the later times, we have

$$\begin{aligned}
 \mathbf{e}_{v,k}(\mathbf{x}) &= \ln (\boldsymbol{\Xi}_k \mathbf{T}_{k-1} \mathbf{T}_k^{-1})^\vee \\
 &= \ln (\boldsymbol{\Xi}_k \exp(\boldsymbol{\epsilon}_{k-1}^\wedge) \mathbf{T}_{\text{op},k-1} \mathbf{T}_{\text{op},k}^{-1} \exp(-\boldsymbol{\epsilon}_k^\wedge))^\vee \\
 &= \ln \underbrace{\left(\boldsymbol{\Xi}_k \mathbf{T}_{\text{op},k-1} \mathbf{T}_{\text{op},k}^{-1} \exp \left((\text{Ad}(\mathbf{T}_{\text{op},k} \mathbf{T}_{\text{op},k-1}^{-1}) \boldsymbol{\epsilon}_{k-1})^\wedge \right) \exp(-\boldsymbol{\epsilon}_k^\wedge) \right)^\vee}_{\exp(\mathbf{e}_{v,k}(\mathbf{x}_{\text{op}})^\wedge)} \\
 &\approx \mathbf{e}_{v,k}(\mathbf{x}_{\text{op}}) + \underbrace{\text{Ad}(\mathbf{T}_{\text{op},k} \mathbf{T}_{\text{op},k-1}^{-1})}_{\mathbf{F}_{k-1}} \boldsymbol{\epsilon}_{k-1} - \boldsymbol{\epsilon}_k,
 \end{aligned} \tag{8.156}$$

where $\mathbf{e}_{v,k}(\mathbf{x}_{\text{op}}) = \ln (\boldsymbol{\Xi}_k \mathbf{T}_{\text{op},k-1} \mathbf{T}_{\text{op},k}^{-1})^\vee$ is the error evaluated at the operating point.

For the measurement errors, we have

$$\begin{aligned}
 \mathbf{e}_{y,jk}(\mathbf{x}) &= \mathbf{y}_{jk} - \mathbf{D}^T \mathbf{T}_k \mathbf{p}_j \\
 &= \mathbf{y}_{jk} - \mathbf{D}^T \exp(\boldsymbol{\epsilon}_k^\wedge) \mathbf{T}_{\text{op},k} \mathbf{p}_j \\
 &\approx \mathbf{y}_{jk} - \mathbf{D}^T (\mathbf{1} + \boldsymbol{\epsilon}_k^\wedge) \mathbf{T}_{\text{op},k} \mathbf{p}_j \\
 &= \underbrace{\mathbf{y}_{jk} - \mathbf{D}^T \mathbf{T}_{\text{op},k} \mathbf{p}_j}_{\mathbf{e}_{y,jk}(\mathbf{x}_{\text{op}})} - \underbrace{\left(\mathbf{D}^T (\mathbf{T}_{\text{op},k} \mathbf{p}_j)^\odot \right)}_{\mathbf{G}_{jk}} \boldsymbol{\epsilon}_k.
 \end{aligned} \tag{8.157}$$

We can stack all of the point measurement errors at time k together so that

$$\mathbf{e}_{y,k}(\mathbf{x}) \approx \mathbf{e}_{y,k}(\mathbf{x}_{\text{op}}) - \mathbf{G}_k \boldsymbol{\epsilon}_k, \tag{8.158}$$

where

$$\mathbf{e}_{y,k}(\mathbf{x}) = \begin{bmatrix} \mathbf{e}_{y,1k}(\mathbf{x}) \\ \vdots \\ \mathbf{e}_{y,Mk}(\mathbf{x}) \end{bmatrix}, \quad \mathbf{e}_{y,k}(\mathbf{x}_{\text{op}}) = \begin{bmatrix} \mathbf{e}_{y,1k}(\mathbf{x}_{\text{op}}) \\ \vdots \\ \mathbf{e}_{y,Mk}(\mathbf{x}_{\text{op}}) \end{bmatrix}, \quad \mathbf{G}_k = \begin{bmatrix} \mathbf{G}_{1k} \\ \vdots \\ \mathbf{G}_{Mk} \end{bmatrix}. \tag{8.159}$$

Next, we will insert these approximations into our objective function to complete the Gauss-Newton derivation.

¹¹ We could also include the $SE(3)$ Jacobian to do better here, as was done in Section 7.3.4, but this is a reasonable starting point.

Gauss-Newton Update

To set up the Gauss-Newton update, we define the following stacked quantities:

$$\delta \mathbf{x} = \begin{bmatrix} \epsilon_0 \\ \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_K \end{bmatrix}, \quad \mathbf{H} = \left[\begin{array}{ccccc} \mathbf{1} & & & & \\ -\mathbf{F}_0 & \mathbf{1} & & & \\ & -\mathbf{F}_1 & \ddots & & \\ & & \ddots & \mathbf{1} & \\ \hline \mathbf{G}_0 & & & -\mathbf{F}_{K-1} & \mathbf{1} \\ & \mathbf{G}_1 & & & \\ & & \mathbf{G}_2 & & \\ & & & \ddots & \\ & & & & \mathbf{G}_K \end{array} \right],$$

$$\mathbf{e}(\mathbf{x}_{\text{op}}) = \begin{bmatrix} \mathbf{e}_{v,0}(\mathbf{x}_{\text{op}}) \\ \mathbf{e}_{v,1}(\mathbf{x}_{\text{op}}) \\ \vdots \\ \mathbf{e}_{v,K-1}(\mathbf{x}_{\text{op}}) \\ \hline \mathbf{e}_{y,0}(\mathbf{x}_{\text{op}}) \\ \mathbf{e}_{y,1}(\mathbf{x}_{\text{op}}) \\ \vdots \\ \mathbf{e}_{y,K-1}(\mathbf{x}_{\text{op}}) \\ \hline \mathbf{e}_{y,K}(\mathbf{x}_{\text{op}}) \end{bmatrix}, \quad (8.160)$$

and

$$\mathbf{W} = \text{diag}(\check{\mathbf{P}}_0, \mathbf{Q}_1, \dots, \mathbf{Q}_K, \mathbf{R}_0, \mathbf{R}_1, \dots, \mathbf{R}_K), \quad (8.161)$$

which are identical in structure to the matrices in the nonlinear version. The quadratic (in terms of the perturbation, $\delta \mathbf{x}$) approximation to the objective function is then

$$J(\mathbf{x}) \approx J(\mathbf{x}_{\text{op}}) - \mathbf{b}^T \delta \mathbf{x} + \frac{1}{2} \delta \mathbf{x}^T \mathbf{A} \delta \mathbf{x}, \quad (8.162)$$

where

$$\mathbf{A} = \underbrace{\mathbf{H}^T \mathbf{W}^{-1} \mathbf{H}}_{\text{block-tridiagonal}}, \quad \mathbf{b} = \mathbf{H}^T \mathbf{W}^{-1} \mathbf{e}(\mathbf{x}_{\text{op}}). \quad (8.163)$$

Minimizing with respect to $\delta \mathbf{x}$, we have

$$\mathbf{A} \delta \mathbf{x}^* = \mathbf{b}, \quad (8.164)$$

for the optimal perturbation,

$$\delta \mathbf{x}^* = \begin{bmatrix} \boldsymbol{\epsilon}_0^* \\ \boldsymbol{\epsilon}_1^* \\ \vdots \\ \boldsymbol{\epsilon}_K^* \end{bmatrix}. \quad (8.165)$$

Once we have the optimal perturbation, we update our operating point through the original perturbation scheme,

$$\mathbf{T}_{\text{op},k} \leftarrow \exp(\boldsymbol{\epsilon}_k^{*\wedge}) \mathbf{T}_{\text{op},k}, \quad (8.166)$$

which ensures that $\mathbf{T}_{\text{op},k}$ stays in $SE(3)$. We then iterate the entire scheme to convergence. As a reminder we note that at the final iteration we have $\hat{\mathbf{T}}_{v_k i} = \mathbf{T}_{\text{op},k}$ as our estimate, but if we prefer we can compute $\hat{\mathbf{T}}_{i v_k} = \hat{\mathbf{T}}_{v_k i}^{-1}$.

Once again, the main concept that we have used to derive this Gauss-Newton optimization problem involving pose variables is to compute the update in the Lie algebra, $\mathfrak{se}(3)$, but store the mean in the Lie group, $SE(3)$.

8.3 Pose-Graph Relaxation

Another classic problem that is worth investigating in our framework is that of *pose-graph relaxation*. Here we do not explicitly measure any points in the stationary frame, but instead begin directly with a set of relative pose ‘measurements’ (a.k.a., pseudomeasurements) that may have come from some form of dead-reckoning. The situation is depicted in Figure 8.3, where we can consider each white triangle to be a reference frame in three dimensions. We refer to this diagram as a *pose graph* in that it only involves poses and no points.

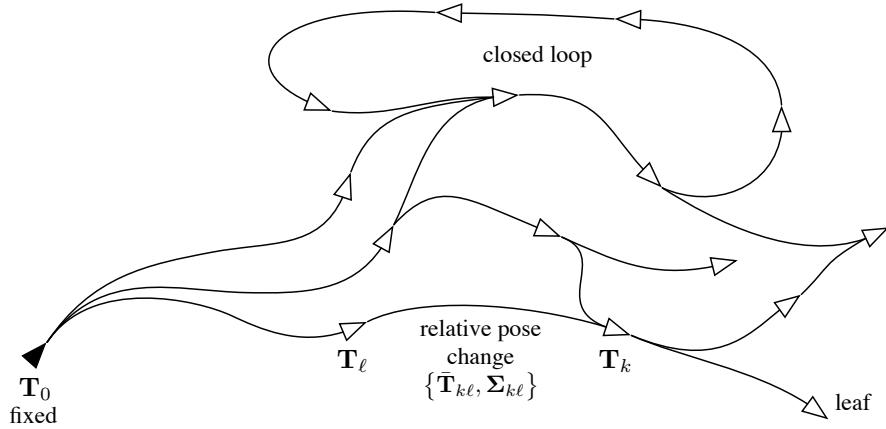
Importantly, pose graphs can contain closed loops (as well as leaf nodes), but unfortunately, the relative pose measurements are uncertain and do not necessarily compound to identity around any loop. Therefore, our task is to ‘relax’ the pose graph with respect to one (arbitrarily selected) pose, called pose 0. In other words, we will determine an optimal estimate for each pose relative to pose 0, given all of the relative pose measurements.

8.3.1 Problem Setup

There is an implicit reference frame, $\underline{\mathcal{F}}_k$, located at pose k in Figure 8.3. We will use a transformation matrix to denote the pose change from $\underline{\mathcal{F}}_0$ to $\underline{\mathcal{F}}_k$:

\mathbf{T}_k : transformation matrix representing pose of $\underline{\mathcal{F}}_k$ relative to $\underline{\mathcal{F}}_0$.

Figure 8.3 In the pose-graph relaxation problem, only relative pose change ‘measurements’ are provided, and the task is to determine where each pose is in relation to one privileged pose, labelled 0, which is fixed. We cannot simply compound the relative transforms due to the presence of closed loops, around which the relative transforms may not compound to identity.



Our task will be to estimate this transformation for all the poses (other than pose 0).

As mentioned above, the measurements will be a set of relative pose changes between nodes in the pose graph. The measurements will be assumed to be Gaussian (on $SE(3)$) and thus have a mean and a covariance given by

$$\{\bar{\mathbf{T}}_{kl}, \Sigma_{kl}\}. \quad (8.167)$$

Explicitly, a random sample, \mathbf{T}_{kl} , is drawn from this Gaussian density according to

$$\mathbf{T}_{kl} = \exp(\hat{\boldsymbol{\xi}}_{kl}) \bar{\mathbf{T}}_{kl}, \quad (8.168)$$

where

$$\boldsymbol{\xi}_{kl} \sim \mathcal{N}(\mathbf{0}, \Sigma_{kl}). \quad (8.169)$$

Measurements of this type can arise from a lower-level dead-reckoning method such as wheel odometry, visual odometry, or inertial sensing. Not all pairs of poses will have relative measurements, making the problem fairly sparse in practice.

8.3.2 Batch Maximum Likelihood Solution

We will follow a batch ML approach very similar to the pose-fusion problem described in Section 7.3.4. As usual, for each measurement we will formulate an error term:

$$\mathbf{e}_{kl}(\mathbf{x}) = \ln \left(\bar{\mathbf{T}}_{kl} (\mathbf{T}_k \mathbf{T}_l^{-1})^{-1} \right)^\vee = \ln \left(\bar{\mathbf{T}}_{kl} \mathbf{T}_l \mathbf{T}_k^{-1} \right)^\vee, \quad (8.170)$$

where we have used the shorthand

$$\mathbf{x} = \{\mathbf{T}_1, \dots, \mathbf{T}_K\}, \quad (8.171)$$

for the state to be estimated. We will adopt the usual $SE(3)$ -sensitive perturbation scheme,

$$\mathbf{T}_k = \exp(\boldsymbol{\epsilon}_k^\wedge) \mathbf{T}_{\text{op},k}, \quad (8.172)$$

where $\mathbf{T}_{\text{op},k}$ is the operating point and $\boldsymbol{\epsilon}_k$ the small perturbation. Inserting this into the error expression, we have

$$\mathbf{e}_{k\ell}(\mathbf{x}) = \ln(\bar{\mathbf{T}}_{k\ell} \exp(\boldsymbol{\epsilon}_\ell^\wedge) \mathbf{T}_{\text{op},\ell} \mathbf{T}_{\text{op},k}^{-1} \exp(-\boldsymbol{\epsilon}_k^\wedge))^\vee. \quad (8.173)$$

We can pull the $\boldsymbol{\epsilon}_\ell$ factor over to the right without approximation:

$$\mathbf{e}_{k\ell}(\mathbf{x}) = \ln \left(\underbrace{\bar{\mathbf{T}}_{k\ell} \mathbf{T}_{\text{op},\ell} \mathbf{T}_{\text{op},k}^{-1}}_{\text{small}} \exp \left((\mathcal{T}_{\text{op},k} \mathcal{T}_{\text{op},\ell}^{-1} \boldsymbol{\epsilon}_\ell)^\wedge \right) \exp(-\boldsymbol{\epsilon}_k^\wedge) \right)^\vee, \quad (8.174)$$

where $\mathcal{T}_{\text{op},k} = \text{Ad}(\mathbf{T}_{\text{op},k})$. Since both $\boldsymbol{\epsilon}_\ell$ and $\boldsymbol{\epsilon}_k$ will be converging toward zero, we can combine these approximately and write

$$\mathbf{e}_{k\ell}(\mathbf{x}) \approx \ln \left(\exp(\mathbf{e}_{k\ell}(\mathbf{x}_{\text{op}})^\wedge) \exp \left((\mathcal{T}_{\text{op},k} \mathcal{T}_{\text{op},\ell}^{-1} \boldsymbol{\epsilon}_\ell - \boldsymbol{\epsilon}_k)^\wedge \right) \right)^\vee, \quad (8.175)$$

where we have also defined

$$\mathbf{e}_{k\ell}(\mathbf{x}_{\text{op}}) = \ln(\bar{\mathbf{T}}_{k\ell} \mathbf{T}_{\text{op},\ell} \mathbf{T}_{\text{op},k}^{-1})^\vee, \quad (8.176a)$$

$$\mathbf{x}_{\text{op}} = \{\mathbf{T}_{\text{op},1}, \dots, \mathbf{T}_{\text{op},K}\}. \quad (8.176b)$$

Finally, we can use the BCH approximation in (7.100) to write our linearized error as

$$\mathbf{e}_{k\ell}(\mathbf{x}) \approx \mathbf{e}_{k\ell}(\mathbf{x}_{\text{op}}) - \mathbf{G}_{k\ell} \delta \mathbf{x}_{k\ell}, \quad (8.177)$$

where

$$\mathbf{G}_{k\ell} = \begin{bmatrix} -\mathcal{J}(-\mathbf{e}_{k\ell}(\mathbf{x}_{\text{op}}))^{-1} \mathcal{T}_{\text{op},k} \mathcal{T}_{\text{op},\ell}^{-1} & \mathcal{J}(-\mathbf{e}_{k\ell}(\mathbf{x}_{\text{op}}))^{-1} \end{bmatrix}, \quad (8.178a)$$

$$\delta \mathbf{x}_{k\ell} = \begin{bmatrix} \boldsymbol{\epsilon}_\ell \\ \boldsymbol{\epsilon}_k \end{bmatrix}. \quad (8.178b)$$

We may choose to approximate $\mathcal{J} \approx \mathbf{1}$ to keep things simple, but as we saw in Section 7.3.4, keeping the full expression has some benefit. This is because even after convergence $\mathbf{e}_{k\ell}(\mathbf{x}_{\text{op}}) \neq \mathbf{0}$; these are the non-zero residual errors for this least-squares problem.

With our linearized error expression in hand, we can now define the ML objective function as

$$J(\mathbf{x}) = \frac{1}{2} \sum_{k,\ell} \mathbf{e}_{k\ell}(\mathbf{x})^T \Sigma_{k\ell}^{-1} \mathbf{e}_{k\ell}(\mathbf{x}), \quad (8.179)$$

where we note that there will be one term in the sum for each relative

pose measurement in the pose graph. Inserting our approximate error expression, we have

$$J(\mathbf{x}) \approx \frac{1}{2} \sum_{k,\ell} \left(\mathbf{e}_{k\ell}(\mathbf{x}_{\text{op}}) - \mathbf{G}_{k\ell} \mathbf{P}_{k\ell} \delta \mathbf{x} \right)^T \boldsymbol{\Sigma}_{k\ell}^{-1} \left(\mathbf{e}_{k\ell}(\mathbf{x}_{\text{op}}) - \mathbf{G}_{k\ell} \mathbf{P}_{k\ell} \delta \mathbf{x} \right), \quad (8.180)$$

or

$$J(\mathbf{x}) \approx J(\mathbf{x}_{\text{op}}) - \mathbf{b}^T \delta \mathbf{x} + \frac{1}{2} \delta \mathbf{x}^T \mathbf{A} \delta \mathbf{x}, \quad (8.181)$$

where

$$\mathbf{b} = \sum_{k,\ell} \mathbf{P}_{k\ell}^T \mathbf{G}_{k\ell}^T \boldsymbol{\Sigma}_{k\ell}^{-1} \mathbf{e}_{k\ell}(\mathbf{x}_{\text{op}}), \quad (8.182a)$$

$$\mathbf{A} = \sum_{k,\ell} \mathbf{P}_{k\ell}^T \mathbf{G}_{k\ell}^T \boldsymbol{\Sigma}_{k\ell}^{-1} \mathbf{G}_{k\ell} \mathbf{P}_{k\ell}, \quad (8.182b)$$

$$\delta \mathbf{x}_{k\ell} = \mathbf{P}_{k\ell} \delta \mathbf{x}, \quad (8.182c)$$

and $\mathbf{P}_{k\ell}$ is a projection matrix to pick out the $k\ell$ th perturbation variables from the full perturbation state,

$$\delta \mathbf{x} = \begin{bmatrix} \boldsymbol{\epsilon}_1 \\ \vdots \\ \boldsymbol{\epsilon}_K \end{bmatrix}. \quad (8.183)$$

Our approximate objective function is now exactly quadratic and we minimize $J(\mathbf{x})$ with respect to $\delta \mathbf{x}$ by taking the derivative:

$$\frac{\partial J(\mathbf{x})}{\partial \delta \mathbf{x}^T} = -\mathbf{b} + \mathbf{A} \delta \mathbf{x} \quad (8.184)$$

Setting this to zero, the optimal perturbation, $\delta \mathbf{x}^*$, is the solution to the following linear system:

$$\mathbf{A} \delta \mathbf{x}^* = \mathbf{b}. \quad (8.185)$$

As usual, the procedure iterates between solving (8.185) for the optimal perturbation,

$$\delta \mathbf{x}^* = \begin{bmatrix} \boldsymbol{\epsilon}_1^* \\ \vdots \\ \boldsymbol{\epsilon}_K^* \end{bmatrix}, \quad (8.186)$$

and updating the nominal quantities using the optimal perturbations according to our original scheme,

$$\mathbf{T}_{\text{op},k} \leftarrow \exp \left(\boldsymbol{\epsilon}_k^* \wedge \right) \mathbf{T}_{\text{op},k}, \quad (8.187)$$

which ensures that $\mathbf{T}_{\text{op},k} \in SE(3)$. We continue until some convergence criterion is met. Once converged, we set $\hat{\mathbf{T}}_k = \mathbf{T}_{\text{op},k}$ at the last iteration as the final estimates for the vehicle poses relative to pose 0.

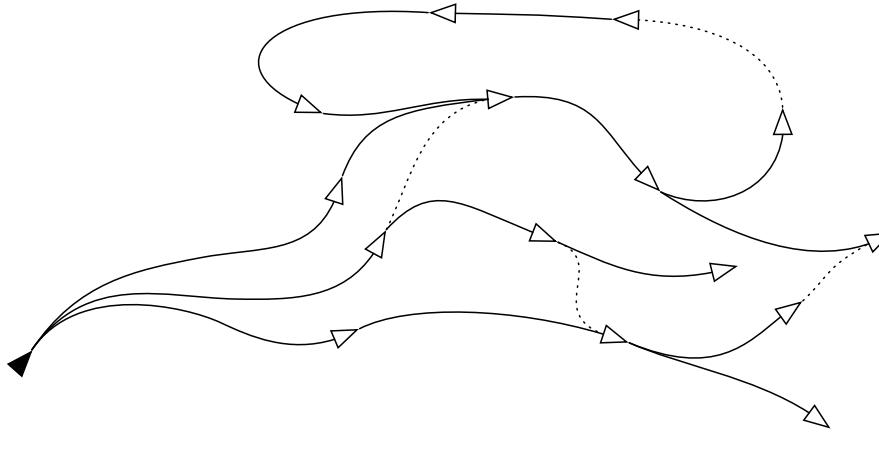


Figure 8.4 The pose-graph relaxation procedure can be initialized by finding a spanning tree (solid lines) within the pose graph. The dotted measurements are discarded (only for initialization) and then all the pose variables are compounded outward from pose 0.

8.3.3 Initialization

There are several ways to initialize the operating point, \mathbf{x}_{op} , at the start of the Gauss-Newton procedure. A common method is to find a spanning tree as in Figure 8.4; the initial values of the pose variables can be found by compounding (a subset of) the relative pose measurements outward from the chosen privileged node 0. Note, the spanning tree is not unique and as such different initialization can be computed. A shallow spanning tree is preferable over a deep one so that as little uncertainty as possible is accumulated to any given node.

8.3.4 Exploiting Sparsity

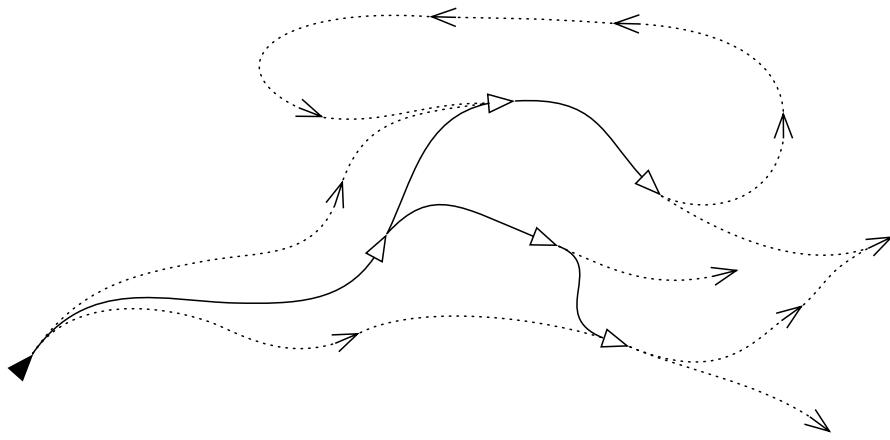
There is inherent sparsity in pose graphs, and this can be exploited to make the pose-graph relaxation procedure more computationally efficient¹². As shown in Figure 8.5, some nodes (shown as open triangles) in the pose graph have either one or two edges, creating two types of *local chains*:

- (i) Constrained: both ends of the chain have a junction (closed-triangle) node and thus the chain's measurements are important to the rest of the pose graph.
- (ii) Cantilevered: only one end of the chain has a junction (closed-triangle) node and thus the chain's measurements do not affect the rest of the pose graph.

We can use any of the pose-compounding methods from Section 7.3.3 to combine the relative pose measurements associated with the constrained local chains and then treat this as a new relative pose measurement that replaces its constituents. Once this is done, we can use

¹² The method in this section should be viewed as an approximation to the brute-force approach in the previous section, owing to the fact that it is a nonlinear system.

Figure 8.5 The pose-graph relaxation method can be sped up by exploiting the inherent sparsity in the pose graph. The open-triangle nodes only have one or two edges and thus do not need to be solved for initially. Instead, the relative measurements passing through the open-triangle nodes (dotted) are combined allowing the closed-triangle nodes to be solved for much more efficiently. The open-triangle nodes also can be solved for afterwards.



the pose-graph relaxation approach to solve for the reduced pose graph formed from only the junction (shown as closed-triangle) nodes.

Afterwards, we can treat all the junction nodes as fixed, and solve for the local chain (open-triangle) nodes. For those nodes in cantilevered local chains, we can simply use one of the pose-compounding methods from Section 7.3.3 to compound outward from the one junction (closed-triangle) node associated with the chain. The cost of this compounding procedure is linear in the length of the local chain (and iteration is not required).

For each constrained local chain, we can run a smaller pose-graph relaxation just for that chain to solve for its nodes. In this case, the two bounding junction (closed-triangle) nodes will be fixed. If we order the variables sequentially along the local chain, the \mathbf{A} matrix for this pose-graph relaxation will be block-tridiagonal and thus the cost of each iteration will be linear in the length of the chain (i.e., sparse Cholesky decomposition followed by forward-backward passes).

This two-phased approach to pose-graph relaxation is not the only way to exploit the inherent sparsity in order to gain computational efficiency. A good sparse solver should be able to exploit the sparsity in the \mathbf{A} matrix for the full system as well, avoiding the need to identify and bookkeep all of the local chains.

8.3.5 Chain Example

It is worthwhile to provide an example of pose-graph relaxation for the short constrained chain in Figure 8.6. We only need to solve for poses 1, 2, 3, and 4 since 0 and 5 are fixed. The \mathbf{A} matrix for this example is

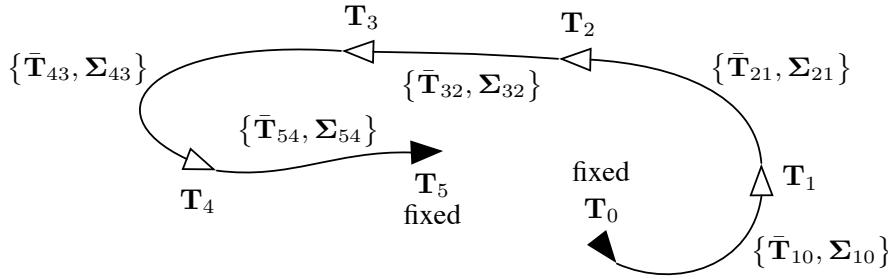


Figure 8.6
Example
pose-graph
relaxation problem
for a constrained
chain of poses.
Here the black
poses are fixed and
we must solve for
the white poses,
given all the
relative pose
measurements.

given by

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & & & \\ \mathbf{A}_{12}^T & \mathbf{A}_{22} & \mathbf{A}_{23} & & \\ & \mathbf{A}_{23}^T & \mathbf{A}_{33} & \mathbf{A}_{34} & \\ & & \mathbf{A}_{34}^T & \mathbf{A}_{44} & \\ \end{bmatrix}$$

$$= \begin{bmatrix} \Sigma_{10}^{-1} + \mathcal{T}_{21}^T \Sigma_{21}^{-1} \mathcal{T}_{21} & -\mathcal{T}_{21}^T \Sigma_{21}^{-1} \mathcal{T}_{21} & & & \\ -\Sigma_{21}^{-1} \mathcal{T}_{21} & \Sigma_{21}^{-1} + \mathcal{T}_{32}^T \Sigma_{32}^{-1} \mathcal{T}_{32} & -\mathcal{T}_{32}^T \Sigma_{32}^{-1} \mathcal{T}_{32} & & \\ & -\Sigma_{32}^{-1} \mathcal{T}_{32} & \Sigma_{32}^{-1} + \mathcal{T}_{43}^T \Sigma_{43}^{-1} \mathcal{T}_{43} & -\mathcal{T}_{43}^T \Sigma_{43}^{-1} \mathcal{T}_{43} & \\ & & -\Sigma_{43}^{-1} \mathcal{T}_{43} & \Sigma_{43}^{-1} + \mathcal{T}_{54}^T \Sigma_{54}^{-1} \mathcal{T}_{54} & \\ \end{bmatrix} \quad (8.188)$$

where

$$\Sigma_{k\ell}^{-1} = \mathcal{J}_{k\ell}^{-T} \Sigma_{k\ell}^{-1} \mathcal{J}_{k\ell}, \quad (8.189a)$$

$$\mathcal{T}_{k\ell} = \mathcal{T}_{\text{op},k} \mathcal{T}_{\text{op},\ell}^{-1}, \quad (8.189b)$$

$$\mathcal{J}_{k\ell} = \mathcal{J}(-\mathbf{e}_{k\ell}(\mathbf{x}_{\text{op}})). \quad (8.189c)$$

The \mathbf{b} matrix is given by

$$\mathbf{b} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \mathbf{b}_3 \\ \mathbf{b}_4 \end{bmatrix} = \begin{bmatrix} \mathcal{J}_{10}^{-T} \Sigma_{10}^{-1} \mathbf{e}_{10}(\mathbf{x}_{\text{op}}) - \mathcal{T}_{21}^T \mathcal{J}_{21}^{-T} \Sigma_{21}^{-1} \mathbf{e}_{21}(\mathbf{x}_{\text{op}}) \\ \mathcal{J}_{21}^{-T} \Sigma_{21}^{-1} \mathbf{e}_{21}(\mathbf{x}_{\text{op}}) - \mathcal{T}_{32}^T \mathcal{J}_{32}^{-T} \Sigma_{32}^{-1} \mathbf{e}_{32}(\mathbf{x}_{\text{op}}) \\ \mathcal{J}_{32}^{-T} \Sigma_{32}^{-1} \mathbf{e}_{32}(\mathbf{x}_{\text{op}}) - \mathcal{T}_{43}^T \mathcal{J}_{43}^{-T} \Sigma_{43}^{-1} \mathbf{e}_{43}(\mathbf{x}_{\text{op}}) \\ \mathcal{J}_{43}^{-T} \Sigma_{43}^{-1} \mathbf{e}_{43}(\mathbf{x}_{\text{op}}) - \mathcal{T}_{54}^T \mathcal{J}_{54}^{-T} \Sigma_{54}^{-1} \mathbf{e}_{54}(\mathbf{x}_{\text{op}}) \end{bmatrix}. \quad (8.190)$$

We can see that for this chain example, \mathbf{A} is block-tridiagonal and we can therefore solve the $\mathbf{A} \delta \mathbf{x}^* = \mathbf{b}$ equation quite efficiently using a sparse Cholesky decomposition as follows. Let

$$\mathbf{A} = \mathbf{U} \mathbf{U}^T, \quad (8.191)$$

where \mathbf{U} is an upper-triangular matrix of the form

$$\mathbf{U} = \begin{bmatrix} \mathbf{U}_{11} & \mathbf{U}_{12} & & \\ & \mathbf{U}_{22} & \mathbf{U}_{23} & \\ & & \mathbf{U}_{33} & \mathbf{U}_{34} \\ & & & \mathbf{U}_{44} \end{bmatrix}. \quad (8.192)$$

The blocks of \mathbf{U} can be solved for as follows:

$$\begin{aligned}
 \mathbf{U}_{44}\mathbf{U}_{44}^T &= \mathbf{A}_{44} : \text{solve for } \mathbf{U}_{44} \text{ using Cholesky decomposition,} \\
 \mathbf{U}_{34}\mathbf{U}_{44}^T &= \mathbf{A}_{34} : \text{solve for } \mathbf{U}_{34} \text{ using linear algebra solver,} \\
 \mathbf{U}_{33}\mathbf{U}_{33}^T + \mathbf{U}_{34}\mathbf{U}_{34}^T &= \mathbf{A}_{33} : \text{solve for } \mathbf{U}_{33} \text{ using Cholesky decomposition,} \\
 \mathbf{U}_{23}\mathbf{U}_{33}^T &= \mathbf{A}_{23} : \text{solve for } \mathbf{U}_{23} \text{ using linear algebra solver,} \\
 \mathbf{U}_{22}\mathbf{U}_{22}^T + \mathbf{U}_{23}\mathbf{U}_{23}^T &= \mathbf{A}_{22} : \text{solve for } \mathbf{U}_{22} \text{ using Cholesky decomposition,} \\
 \mathbf{U}_{12}\mathbf{U}_{22}^T &= \mathbf{A}_{12} : \text{solve for } \mathbf{U}_{12} \text{ using linear algebra solver,} \\
 \mathbf{U}_{11}\mathbf{U}_{11}^T + \mathbf{U}_{12}\mathbf{U}_{12}^T &= \mathbf{A}_{11} : \text{solve for } \mathbf{U}_{11} \text{ using Cholesky decomposition.}
 \end{aligned}$$

Then we can carry out a backward pass followed by a forward pass to solve for $\delta\mathbf{x}^*$:

$$\begin{array}{ll}
 \text{backward pass} & \text{forward pass} \\
 \mathbf{U}\mathbf{c} = \mathbf{b} & \mathbf{U}^T \delta\mathbf{x}^* = \mathbf{c}
 \end{array}$$

$$\begin{array}{ll}
 \mathbf{U}_{44}\mathbf{c}_4 = \mathbf{b}_4 & \mathbf{U}_{11}^T \boldsymbol{\epsilon}_1^* = \mathbf{c}_1 \\
 \mathbf{U}_{33}\mathbf{c}_3 + \mathbf{U}_{34}\mathbf{c}_4 = \mathbf{b}_3 & \mathbf{U}_{12}^T \boldsymbol{\epsilon}_1^* + \mathbf{U}_{22}^T \boldsymbol{\epsilon}_2^* = \mathbf{c}_2 \\
 \mathbf{U}_{22}\mathbf{c}_2 + \mathbf{U}_{23}\mathbf{c}_3 = \mathbf{b}_2 & \mathbf{U}_{23}^T \boldsymbol{\epsilon}_2^* + \mathbf{U}_{33}^T \boldsymbol{\epsilon}_3^* = \mathbf{c}_3 \\
 \mathbf{U}_{11}\mathbf{c}_1 + \mathbf{U}_{12}\mathbf{c}_2 = \mathbf{b}_1 & \mathbf{U}_{34}^T \boldsymbol{\epsilon}_3^* + \mathbf{U}_{44}^T \boldsymbol{\epsilon}_4^* = \mathbf{c}_4
 \end{array}$$

First we proceed down the left column solving for \mathbf{c}_4 , \mathbf{c}_3 , \mathbf{c}_2 , and \mathbf{c}_1 . Then we proceed down the right column solving for $\boldsymbol{\epsilon}_1^*$, $\boldsymbol{\epsilon}_2^*$, $\boldsymbol{\epsilon}_3^*$, and $\boldsymbol{\epsilon}_4^*$. The cost of solving for each of \mathbf{U} , \mathbf{c} , and finally $\delta\mathbf{x}^*$ is linear in the length of the chain. Once we solve for $\delta\mathbf{x}^*$, we update the operating points of each pose variable,

$$\mathbf{T}_{\text{op},k} \leftarrow \exp\left(\boldsymbol{\epsilon}_k^* \wedge\right) \mathbf{T}_{\text{op},k}, \quad (8.193)$$

and iterate the entire procedure to convergence. For this short chain, exploiting the sparsity may not be worthwhile, but for very long constrained chains the benefits are obvious.

9

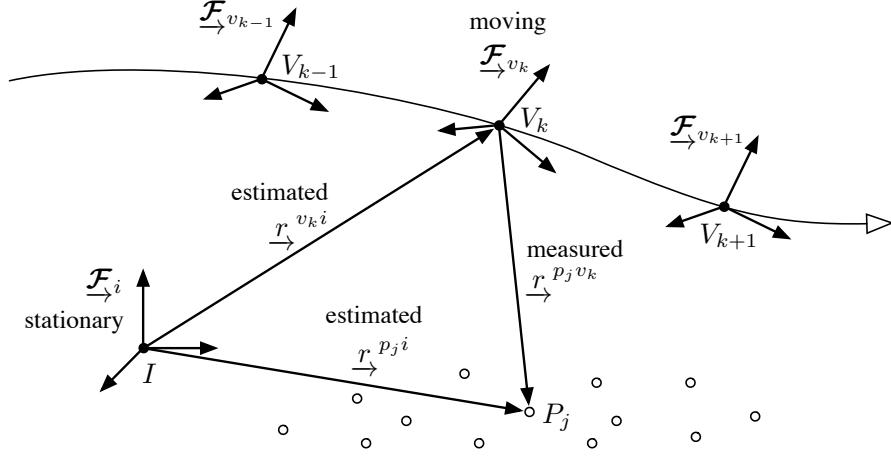
Pose-and-Point Estimation Problems

In this chapter of the book, we will address one of the most fundamental problems in mobile robotics, estimating the trajectory of a robot and the structure of the world around it (i.e., point landmarks) together. In robotics, this is called the *simultaneous localization and mapping (SLAM)* problem. However, in computer vision an almost identical problem came to prominence through the application of aligning aerial photographs into a mosaic; the classic solution to this problem is called *bundle adjustment (BA)*. We will look at BA through the lens of our $SE(3)$ estimation techniques.

9.1 Bundle Adjustment

Photogrammetry, the process of aerial map building, has been in use since the 1920s (Dyce, 2013). It involves flying an airplane along a route, taking hundreds or thousands of pictures of the terrain below, and then stitching these together into a mosaic. In the early days, photogrammetry was a highly laborious process; printed photographs were spread out on a large surface and aligned by hand. From the late 1920s until the 1960s, clever projectors called *multiplex stereoplotters* were used to more precisely align photos, but it was still a painstaking, manual process. The first automated stitching of photos occurred in the 1960s with the advent of computers and an algorithm called bundle adjustment (Brown, 1958). Starting around the 1970s, aerial map making was gradually replaced by satellite-based mapping (e.g., the US Landsat program), but the basic algorithms for image stitching remain the same (Triggs et al., 2000). It is worth noting that the robustness of automated photogrammetry was increased significantly with the invention of modern feature detectors in computer vision, starting with the work of Lowe (2004). Today, commercial software packages exist that automate the photogrammetry process well, and they all essentially use BA for alignment.

Figure 9.1
 Definition of reference frames for the bundle adjustment problem. There is a stationary reference frame and a moving reference frame, attached to a vehicle. A collection of points, P_j , are observed by the moving vehicle (using a camera) and the goal is to determine the relative pose of the moving frame with respect to the stationary one (at all of the times of interest) as well as the positions of all of the points in the stationary frame.



9.1.1 Problem Setup

Figure 9.1 shows the setup for our bundle adjustment problem. The state that we wish to estimate is

$\mathbf{T}_k = \mathbf{T}_{v_k i}$: transformation matrix representing the pose of the vehicle at time k

$\mathbf{p}_j = \begin{bmatrix} \mathbf{r}_i^{p_j i} \\ 1 \end{bmatrix}$: homogeneous point representing the position of landmark j

where $k = 1 \dots K$ and $j = 1 \dots M$ and we will use the cleaner version of the notation to avoid writing out all the sub- and super-scripts throughout the derivation. We will use the shorthand,

$$\mathbf{x} = \{\mathbf{T}_1, \dots, \mathbf{T}_K, \mathbf{p}_1, \dots, \mathbf{p}_M\}, \quad (9.1)$$

to indicate the entire state that we wish to estimate as well as $\mathbf{x}_{jk} = \{\mathbf{T}_k, \mathbf{p}_j\}$ to indicate the subset of the state including the k th pose and j th landmark. Notably, we exclude \mathbf{T}_0 from the state to be estimated as the system is otherwise unobservable; recall the discussion in Section 5.1.3 about unknown measurement bias.

9.1.2 Measurement Model

There are two main differences between the problem treated here and the one from the previous chapter. First, we are now estimating the point positions in addition to the poses. Second, we will introduce a nonlinear sensor model (e.g., a camera) such that we have a more complicated measurement than simply the point expressed in the vehicle frame.

Nonlinear Model

The measurement, \mathbf{y}_{jk} , will correspond to some observation of point j from pose k (i.e., some function of $\mathbf{r}_{v_k}^{p_j v_k}$). The measurement model for this problem will be of the form

$$\mathbf{y}_{jk} = \mathbf{g}(\mathbf{x}_{jk}) + \mathbf{n}_{jk}, \quad (9.2)$$

where $\mathbf{g}(\cdot)$ is the nonlinear model and $\mathbf{n}_{jk} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_{jk})$ is additive Gaussian noise. We can use the shorthand

$$\mathbf{y} = \{\mathbf{y}_{10}, \dots, \mathbf{y}_{M0}, \dots, \mathbf{y}_{1K}, \dots, \mathbf{y}_{MK}\}, \quad (9.3)$$

to capture all the measurements that we have available.

As discussed in Section 7.3.5, we can think of the overall observation model as the composition of two nonlinearities: one to transform the point into the vehicle frame and one to turn that point into the actual sensor measurement through a camera (or other sensor) model. Letting

$$\mathbf{z}(\mathbf{x}_{jk}) = \mathbf{T}_k \mathbf{p}_j, \quad (9.4)$$

we can write

$$\mathbf{g}(\mathbf{x}_{jk}) = \mathbf{s}(\mathbf{z}(\mathbf{x}_{jk})), \quad (9.5)$$

where $\mathbf{s}(\cdot)$ is the nonlinear camera model¹. In other words, we have $\mathbf{g} = \mathbf{s} \circ \mathbf{z}$, in terms of the composition of functions.

Perturbed Model

We will go one step beyond simply linearizing our model and work out the perturbed model to second order. This could be used, for example, to estimate the bias in using ML estimation, as discussed to Section 4.3.3.

We define the following perturbations to our state variables:

$$\mathbf{T}_k = \exp(\boldsymbol{\epsilon}_k^\wedge) \mathbf{T}_{\text{op},k} \approx \left(\mathbf{1} + \boldsymbol{\epsilon}_k^\wedge + \frac{1}{2} \boldsymbol{\epsilon}_k^\wedge \boldsymbol{\epsilon}_k^\wedge \right) \mathbf{T}_{\text{op},k}, \quad (9.6a)$$

$$\mathbf{p}_j = \mathbf{p}_{\text{op},j} + \mathbf{D} \boldsymbol{\zeta}_j, \quad (9.6b)$$

where

$$\mathbf{D} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \quad (9.7)$$

is a dilation matrix so that our landmark perturbation, $\boldsymbol{\zeta}_j$, is 3×1 . We will use the shorthand $\mathbf{x}_{\text{op}} = \{\mathbf{T}_{\text{op},1}, \dots, \mathbf{T}_{\text{op},K}, \mathbf{p}_{\text{op},1}, \dots, \mathbf{p}_{\text{op},M}\}$ to indicate the entire trajectory's linearization operating point as well as $\mathbf{x}_{\text{op},jk} = \{\mathbf{T}_{\text{op},k}, \mathbf{p}_{\text{op},j}\}$ to indicate the subset of the operating point

¹ See Section 6.4 for several possibilities for the camera (or sensor) model.

including the k th pose and j th landmark. The perturbations will be denoted

$$\delta \mathbf{x} = \begin{bmatrix} \boldsymbol{\epsilon}_1 \\ \vdots \\ \frac{\boldsymbol{\epsilon}_K}{\zeta_1} \\ \vdots \\ \zeta_M \end{bmatrix}, \quad (9.8)$$

with the pose quantities on top and the landmark quantities on the bottom. We will also use

$$\delta \mathbf{x}_{jk} = \begin{bmatrix} \boldsymbol{\epsilon}_k \\ \zeta_j \end{bmatrix} \quad (9.9)$$

to indicate just the perturbations associated with the k th pose and the j th landmark.

Using the perturbation schemes above, we have that

$$\begin{aligned} \mathbf{z}(\mathbf{x}_{jk}) &\approx \left(\mathbf{1} + \boldsymbol{\epsilon}_k^\wedge + \frac{1}{2} \boldsymbol{\epsilon}_k^\wedge \boldsymbol{\epsilon}_k^\wedge \right) \mathbf{T}_{\text{op},k} (\mathbf{p}_{\text{op},j} + \mathbf{D} \zeta_j) \\ &\approx \mathbf{T}_{\text{op},k} \mathbf{p}_{\text{op},j} + \boldsymbol{\epsilon}_k^\wedge \mathbf{T}_{\text{op},k} \mathbf{p}_{\text{op},j} + \mathbf{T}_{\text{op},k} \mathbf{D} \zeta_j \\ &\quad + \frac{1}{2} \boldsymbol{\epsilon}_k^\wedge \boldsymbol{\epsilon}_k^\wedge \mathbf{T}_{\text{op},k} \mathbf{p}_{\text{op},j} + \boldsymbol{\epsilon}_k^\wedge \mathbf{T}_{\text{op},k} \mathbf{D} \zeta_j \\ &= \mathbf{z}(\mathbf{x}_{\text{op},jk}) + \mathbf{Z}_{jk} \delta \mathbf{x}_{jk} + \frac{1}{2} \sum_i \mathbf{1}_i \underbrace{\delta \mathbf{x}_{jk}^T \mathcal{Z}_{ijk} \delta \mathbf{x}_{jk}}_{\text{scalar}}, \end{aligned} \quad (9.10)$$

correct to second order, where

$$\mathbf{z}(\mathbf{x}_{\text{op},jk}) = \mathbf{T}_{\text{op},k} \mathbf{p}_{\text{op},j}, \quad (9.11a)$$

$$\mathbf{Z}_{jk} = [(\mathbf{T}_{\text{op},k} \mathbf{p}_{\text{op},j})^\odot \quad \mathbf{T}_{\text{op},k} \mathbf{D}], \quad (9.11b)$$

$$\mathcal{Z}_{ijk} = \begin{bmatrix} \mathbf{1}_i^\odot (\mathbf{T}_{\text{op},k} \mathbf{p}_{\text{op},j})^\odot & \mathbf{1}_i^\odot \mathbf{T}_{\text{op},k} \mathbf{D} \\ (\mathbf{1}_i^\odot \mathbf{T}_{\text{op},k} \mathbf{D})^T & \mathbf{0} \end{bmatrix}, \quad (9.11c)$$

and i is an index over the rows of $\mathbf{z}(\cdot)$, and $\mathbf{1}_i$ is the i th column of the identity matrix, $\mathbf{1}$.

To then apply the nonlinear camera model, we use the chain rule (for

first and second derivatives), so that

$$\begin{aligned}
\mathbf{g}(\mathbf{x}_{jk}) &= \mathbf{s}(\mathbf{z}(\mathbf{x}_{jk})) \\
&\approx \mathbf{s}\left(\underbrace{\mathbf{z}_{\text{op},jk} + \mathbf{Z}_{jk} \delta \mathbf{x}_{jk} + \frac{1}{2} \sum_m \mathbf{1}_m \delta \mathbf{x}_{jk}^T \mathcal{Z}_{mjk} \delta \mathbf{x}_{jk}}_{\delta \mathbf{z}_{jk}}\right) \\
&\approx \mathbf{s}(\mathbf{z}_{\text{op},jk}) + \mathbf{S}_{jk} \delta \mathbf{z}_{jk} + \frac{1}{2} \sum_i \mathbf{1}_i^T \delta \mathbf{z}_{jk}^T \mathcal{S}_{ijk} \delta \mathbf{z}_{jk} \\
&= \mathbf{s}(\mathbf{z}_{\text{op},jk}) \\
&\quad + \sum_i \mathbf{1}_i (\mathbf{1}_i^T \mathbf{S}_{jk}) \left(\mathbf{Z}_{jk} \delta \mathbf{x}_{jk} + \frac{1}{2} \sum_m \mathbf{1}_m \delta \mathbf{x}_{jk}^T \mathcal{Z}_{mjk} \delta \mathbf{x}_{jk} \right) \\
&\quad + \frac{1}{2} \sum_i \mathbf{1}_i \left(\mathbf{Z}_{jk} \delta \mathbf{x}_{jk} + \frac{1}{2} \sum_m \mathbf{1}_m \delta \mathbf{x}_{jk}^T \mathcal{Z}_{mjk} \delta \mathbf{x}_{jk} \right)^T \\
&\quad \times \mathcal{S}_{ijk} \left(\mathbf{Z}_{jk} \delta \mathbf{x}_{jk} + \frac{1}{2} \sum_m \mathbf{1}_m \delta \mathbf{x}_{jk}^T \mathcal{Z}_{mjk} \delta \mathbf{x}_{jk} \right) \\
&\approx \mathbf{g}(\mathbf{x}_{\text{op},jk}) + \mathbf{G}_{jk} \delta \mathbf{x}_{jk} + \frac{1}{2} \sum_i \mathbf{1}_i \underbrace{\delta \mathbf{x}_{jk}^T \mathcal{G}_{ijk} \delta \mathbf{x}_{jk}}_{\text{scalar}}, \tag{9.12}
\end{aligned}$$

correct to second order, where

$$\mathbf{g}(\mathbf{x}_{\text{op},jk}) = \mathbf{s}(\mathbf{z}(\mathbf{x}_{\text{op},jk})), \tag{9.13a}$$

$$\mathbf{G}_{jk} = \mathbf{S}_{jk} \mathbf{Z}_{jk}, \tag{9.13b}$$

$$\mathbf{S}_{jk} = \left. \frac{\partial \mathbf{s}}{\partial \mathbf{z}} \right|_{\mathbf{z}(\mathbf{x}_{\text{op},jk})}, \tag{9.13c}$$

$$\mathcal{G}_{ijk} = \mathbf{Z}_{jk}^T \mathcal{S}_{ijk} \mathbf{Z}_{jk} + \sum_m \underbrace{\mathbf{1}_i^T \mathbf{S}_{jk} \mathbf{1}_m}_{\text{scalar}} \mathcal{Z}_{mjk}, \tag{9.13d}$$

$$\mathcal{S}_{ijk} = \left. \frac{\partial^2 s_i}{\partial \mathbf{z} \partial \mathbf{z}^T} \right|_{\mathbf{z}(\mathbf{x}_{\text{op},jk})}, \tag{9.13e}$$

and i is an index over the rows of $\mathbf{s}(\cdot)$, and $\mathbf{1}_i$ is the i th column of the identity matrix, $\mathbf{1}$.

If we only care about the linearized (i.e., first-order) model, then we can simply use

$$\mathbf{g}(\mathbf{x}_{jk}) \approx \mathbf{g}(\mathbf{x}_{\text{op},jk}) + \mathbf{G}_{jk} \delta \mathbf{x}_{jk}, \tag{9.14}$$

for our approximate observation model.

9.1.3 Maximum Likelihood Solution

We will set up the bundle adjustment problem using the ML framework described in Section 4.3.3, which means we will not use a motion prior².

For each observation of a point from a pose, we define an error term as

$$\mathbf{e}_{y,jk}(\mathbf{x}) = \mathbf{y}_{jk} - \mathbf{g}(\mathbf{x}_{jk}), \quad (9.15)$$

where \mathbf{y}_{jk} is the measured quantity and \mathbf{g} is our observation model described above. We seek to find the values of \mathbf{x} to minimize the following objective function:

$$J(\mathbf{x}) = \frac{1}{2} \sum_{j,k} \mathbf{e}_{y,jk}(\mathbf{x})^T \mathbf{R}_{jk}^{-1} \mathbf{e}_{y,jk}(\mathbf{x}), \quad (9.16)$$

where \mathbf{x} is the full state that we wish to estimate (all poses and landmarks) and \mathbf{R}_{jk} is the symmetric, positive-definite covariance matrix associated with the jk th measurement. If a particular landmark is not actually observed from a particular pose, we can simply delete the appropriate term from the objective function. The usual approach to this estimation problem is to apply the Gauss-Newton method. Here we will derive the full Newton's method and then approximate to arrive at Gauss-Newton.

Newton's Method

Approximating the error function, we have

$$\mathbf{e}_{y,jk}(\mathbf{x}) \approx \underbrace{\mathbf{y}_{jk} - \mathbf{g}(\mathbf{x}_{op,jk})}_{\mathbf{e}_{y,jk}(\mathbf{x}_{op})} - \mathbf{G}_{jk} \delta \mathbf{x}_{jk} - \frac{1}{2} \sum_i \mathbf{1}_i \delta \mathbf{x}_{jk}^T \mathbf{G}_{ijk} \delta \mathbf{x}_{jk}, \quad (9.17)$$

and thus for the perturbed objective function, we have

$$J(\mathbf{x}) \approx J(\mathbf{x}_{op}) - \mathbf{b}^T \delta \mathbf{x} + \frac{1}{2} \delta \mathbf{x}^T \mathbf{A} \delta \mathbf{x}, \quad (9.18)$$

correct to second order, where

$$\mathbf{b} = \sum_{j,k} \mathbf{P}_{jk}^T \mathbf{G}_{jk}^T \mathbf{R}_{jk}^{-1} \mathbf{e}_{y,jk}(\mathbf{x}_{op}), \quad (9.19a)$$

$$\mathbf{A} = \sum_{j,k} \mathbf{P}_{jk}^T \left(\mathbf{G}_{jk}^T \mathbf{R}_{jk}^{-1} \mathbf{G}_{jk} - \underbrace{\sum_i \mathbf{1}_i^T \mathbf{R}_{jk}^{-1} \mathbf{e}_{y,jk}(\mathbf{x}_{op}) \mathbf{G}_{ijk}}_{\text{scalar}} \right) \mathbf{P}_{jk}, \quad (9.19b)$$

$$\delta \mathbf{x}_{jk} = \mathbf{P}_{jk} \delta \mathbf{x}, \quad (9.19c)$$

² In robotics, when a motion prior or odometry smoothing terms are introduced, we typically call this SLAM.

where \mathbf{P}_{jk} is an appropriate projection matrix to pick off the jk th components of the overall perturbed state, $\delta\mathbf{x}$.

It is worth noting that \mathbf{A} is symmetric, positive-definite. We can see the term that Gauss-Newton normally neglects in the Hessian of J . When $\mathbf{e}_{y,jk}(\mathbf{x}_{op})$ is small, this new term has little effect (and this is the usual justification for its neglect). However, far from the minimum, this term will be more significant and could improve the rate and region of convergence³. We will consider the Gauss-Newton approximation in the next section.

We now minimize $J(\mathbf{x})$ with respect to $\delta\mathbf{x}$ by taking the derivative:

$$\frac{\partial J(\mathbf{x})}{\partial \delta\mathbf{x}^T} = -\mathbf{b} + \mathbf{A} \delta\mathbf{x}. \quad (9.20)$$

Setting this to zero, the optimal perturbation, $\delta\mathbf{x}^*$, is the solution to the following linear system:

$$\mathbf{A} \delta\mathbf{x}^* = \mathbf{b}. \quad (9.21)$$

As usual, the procedure iterates between solving (9.21) for the optimal perturbation,

$$\delta\mathbf{x}^* = \begin{bmatrix} \boldsymbol{\epsilon}_1^* \\ \vdots \\ \frac{\boldsymbol{\epsilon}_K^*}{\zeta_1^*} \\ \vdots \\ \zeta_M^* \end{bmatrix}, \quad (9.22)$$

and updating the nominal quantities using the optimal perturbations according to our original schemes,

$$\mathbf{T}_{op,k} \leftarrow \exp\left(\boldsymbol{\epsilon}_k^*\right) \mathbf{T}_{op,k}, \quad (9.23a)$$

$$\mathbf{p}_{op,j} \leftarrow \mathbf{p}_{op,j} + \mathbf{D} \zeta_j^*, \quad (9.23b)$$

which ensure that $\mathbf{T}_{op,k} \in SE(3)$ and $\mathbf{p}_{op,j}$ keeps its bottom (fourth) entry equal to 1. We continue until some convergence criterion is met. Once converged, we set $\hat{\mathbf{T}}_{v_k i} = \mathbf{T}_{op,k}$ and $\hat{\mathbf{p}}_i^{p_j i} = \mathbf{p}_{op,j}$ at the last iteration as the final estimates for the vehicle poses and landmark positions of interest.

Gauss-Newton Method

Typically in practice, the Gauss-Newton approximation to the Hessian is taken so that at each iteration we solve the linear system

$$\mathbf{A} \delta\mathbf{x}^* = \mathbf{b}, \quad (9.24)$$

³ In practice, including this extra term sometimes makes the numerical stability of the whole procedure worse, so it should be added with caution.

with

$$\mathbf{b} = \sum_{j,k} \mathbf{P}_{jk}^T \mathbf{G}_{jk}^T \mathbf{R}_{jk}^{-1} \mathbf{e}_{y,jk}(\mathbf{x}_{\text{op}}), \quad (9.25a)$$

$$\mathbf{A} = \sum_{j,k} \mathbf{P}_{jk}^T \mathbf{G}_{jk}^T \mathbf{R}_{jk}^{-1} \mathbf{G}_{jk} \mathbf{P}_{jk}, \quad (9.25b)$$

$$\delta \mathbf{x}_{jk} = \mathbf{P}_{jk} \delta \mathbf{x}. \quad (9.25c)$$

This has the significant advantage of not requiring the second derivative of the measurement model to be computed. Assembling the linear system, we find it has the form

$$\underbrace{\mathbf{G}^T \mathbf{R}^{-1} \mathbf{G}}_{\mathbf{A}} \delta \mathbf{x}^* = \underbrace{\mathbf{G}^T \mathbf{R}^{-1} \mathbf{e}_y(\mathbf{x}_{\text{op}})}_{\mathbf{b}}, \quad (9.26)$$

with

$$\begin{aligned} \mathbf{G}_{jk} &= [\mathbf{G}_{1,jk} \quad \mathbf{G}_{2,jk}], \\ \mathbf{G}_{1,jk} &= \mathbf{S}_{jk} (\mathbf{T}_{\text{op},k} \mathbf{p}_{\text{op},j})^\odot, \quad \mathbf{G}_{2,jk} = \mathbf{S}_{jk} \mathbf{T}_{\text{op},k} \mathbf{D}, \end{aligned} \quad (9.27)$$

using the definitions from earlier.

In the case of $K = 3$ free poses (plus fixed pose 0) and $M = 2$ landmarks, the matrices have the form

$$\begin{aligned} \mathbf{G} = [\mathbf{G}_1 \mid \mathbf{G}_2] &= \left[\begin{array}{c|ccccc} & & & & \mathbf{G}_{2,10} & \mathbf{G}_{2,20} \\ \mathbf{G}_{1,11} & & & & \mathbf{G}_{2,11} & \mathbf{G}_{2,21} \\ \mathbf{G}_{1,21} & & \mathbf{G}_{1,12} & & \mathbf{G}_{2,12} & \mathbf{G}_{2,22} \\ & \mathbf{G}_{1,22} & & & \mathbf{G}_{2,13} & \mathbf{G}_{2,23} \\ & & \mathbf{G}_{1,13} & & & \\ & & \mathbf{G}_{1,23} & & & \end{array} \right], \\ \mathbf{e}_y(\mathbf{x}_{\text{op}}) &= \begin{bmatrix} \mathbf{e}_{y,10}(\mathbf{x}_{\text{op}}) \\ \mathbf{e}_{y,20}(\mathbf{x}_{\text{op}}) \\ \mathbf{e}_{y,11}(\mathbf{x}_{\text{op}}) \\ \mathbf{e}_{y,21}(\mathbf{x}_{\text{op}}) \\ \mathbf{e}_{y,12}(\mathbf{x}_{\text{op}}) \\ \mathbf{e}_{y,22}(\mathbf{x}_{\text{op}}) \\ \mathbf{e}_{y,13}(\mathbf{x}_{\text{op}}) \\ \mathbf{e}_{y,23}(\mathbf{x}_{\text{op}}) \end{bmatrix}, \\ \mathbf{R} &= \text{diag}(\mathbf{R}_{10}, \mathbf{R}_{20}, \mathbf{R}_{11}, \mathbf{R}_{21}, \mathbf{R}_{12}, \mathbf{R}_{22}, \mathbf{R}_{13}, \mathbf{R}_{23}), \end{aligned} \quad (9.28)$$

under one particular ordering of the measurements.

In general, multiplying out the left-hand side, $\mathbf{A} = \mathbf{G}^T \mathbf{R}^{-1} \mathbf{G}$, we see that

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{12}^T & \mathbf{A}_{22} \end{bmatrix}, \quad (9.29)$$

where

$$\mathbf{A}_{11} = \mathbf{G}_1^T \mathbf{R}^{-1} \mathbf{G}_1 = \text{diag}(\mathbf{A}_{11,1}, \dots, \mathbf{A}_{11,K}), \quad (9.30\text{a})$$

$$\mathbf{A}_{11,k} = \sum_{j=1}^M \mathbf{G}_{1,jk}^T \mathbf{R}_{jk}^{-1} \mathbf{G}_{1,jk}, \quad (9.30\text{b})$$

$$\mathbf{A}_{12} = \mathbf{G}_1^T \mathbf{R}^{-1} \mathbf{G}_2 = \begin{bmatrix} \mathbf{A}_{12,11} & \cdots & \mathbf{A}_{12,M1} \\ \vdots & \ddots & \vdots \\ \mathbf{A}_{12,1K} & \cdots & \mathbf{A}_{12,MK} \end{bmatrix}, \quad (9.30\text{c})$$

$$\mathbf{A}_{12,jk} = \mathbf{G}_{1,jk}^T \mathbf{R}_{jk}^{-1} \mathbf{G}_{2,jk}, \quad (9.30\text{d})$$

$$\mathbf{A}_{22} = \mathbf{G}_2^T \mathbf{R}^{-1} \mathbf{G}_2 = \text{diag}(\mathbf{A}_{22,1}, \dots, \mathbf{A}_{22,M}), \quad (9.30\text{e})$$

$$\mathbf{A}_{22,j} = \sum_{k=0}^K \mathbf{G}_{2,jk}^T \mathbf{R}_{jk}^{-1} \mathbf{G}_{2,jk}. \quad (9.30\text{f})$$

The fact that both \mathbf{A}_{11} and \mathbf{A}_{22} are block-diagonal means this system has a very special sparsity pattern that can be exploited to efficiently solve for $\delta\mathbf{x}^*$ at each iteration. This will be discussed in detail in the next section.

9.1.4 Exploiting Sparsity

Whether we choose to use Newton's method or Gauss-Newton, we are faced with solving a system of the following form at each iteration:

$$\underbrace{\begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{12}^T & \mathbf{A}_{22} \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} \delta\mathbf{x}_1^* \\ \delta\mathbf{x}_2^* \end{bmatrix}}_{\delta\mathbf{x}^*} = \underbrace{\begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix}}_{\mathbf{b}}, \quad (9.31)$$

where the state, $\delta\mathbf{x}^*$, has been partitioned into parts corresponding to (1) the pose perturbation, $\delta\mathbf{x}_1^* = \boldsymbol{\epsilon}^*$, and (2) the landmark perturbations, $\delta\mathbf{x}_2^* = \boldsymbol{\zeta}^*$.

It turns out that the Hessian of the objective function, \mathbf{A} , has a very special sparsity pattern as depicted in Figure 9.2; it is sometimes referred to as an *arrowhead* matrix. This pattern is due to the presence of the projection matrices, \mathbf{P}_{jk} , in each term of \mathbf{A} ; they embody the fact that each measurement involves just one pose variable and one landmark.

As seen in Figure 9.2, we have that \mathbf{A}_{11} and \mathbf{A}_{22} are both block-diagonal because each measurement involves only one pose and one landmark at a time. We can exploit this sparsity to efficiently solve (9.21) for $\delta\mathbf{x}^*$; this is sometimes referred to as *sparse bundle adjustment*. There are few different ways to do this; we will discuss the Schur complement and a Cholesky technique.

Figure 9.2
 Sparsity pattern of \mathbf{A} . Non-zero entries are indicated by *. This structure is often referred to as an *arrowhead* matrix, because the ζ part is large compared to the ϵ part.

$$\mathbf{A} = \begin{bmatrix} * & & & & & & & & \\ * & * & * & * & * & * & * & * & \epsilon_1 \\ * & * & * & * & * & * & * & * & \vdots \\ * & * & * & * & * & * & * & * & \epsilon_k \\ \ddots & \vdots \\ * & * & * & * & * & * & * & * & \epsilon_K \\ * & * & * & * & * & * & * & * & \zeta_1 \\ * & * & * & * & * & * & * & * & \vdots \\ * & * & * & * & * & * & * & * & \zeta_j \\ \vdots & \vdots \\ * & * & * & * & * & * & * & * & \zeta_M \\ * & * & * & * & * & * & * & * & \epsilon_1 \dots \epsilon_k \dots \epsilon_K \zeta_1 \dots \zeta_j \dots \zeta_M \end{bmatrix}$$

Schur Complement

Typically, the Schur complement is used to manipulate (9.31) into a form that is more efficiently solved. This can be seen by premultiplying both sides by

$$\begin{bmatrix} \mathbf{1} & -\mathbf{A}_{12}\mathbf{A}_{22}^{-1} \\ \mathbf{0} & \mathbf{1} \end{bmatrix},$$

so that

$$\begin{bmatrix} \mathbf{A}_{11} - \mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{12}^T & \mathbf{0} \\ \mathbf{A}_{12}^T & \mathbf{A}_{22} \end{bmatrix} \begin{bmatrix} \delta\mathbf{x}_1^* \\ \delta\mathbf{x}_2^* \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 - \mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{b}_2 \\ \mathbf{b}_2 \end{bmatrix},$$

which has the same solution as (9.31). We may then easily solve for $\delta\mathbf{x}_1^*$ and since \mathbf{A}_{22} is block-diagonal, \mathbf{A}_{22}^{-1} is cheap to compute. Finally, $\delta\mathbf{x}_2^*$ (if desired) can also be efficiently computed through back-substitution, again owing to the sparsity of \mathbf{A}_{22} . This procedure brings the complexity of each solve down from $O((K+M)^3)$ without sparsity to $O(K^3 + K^2M)$ with sparsity, which is most beneficial when $K \ll M$.

A similar procedure can be had by exploiting the sparsity of \mathbf{A}_{11} , but in robotics problems we may also have some additional measurements that perturb this structure and, more importantly, $\delta\mathbf{x}_2^*$ is usually much larger than $\delta\mathbf{x}_1^*$ in bundle adjustment. While the Schur complement method works well, it does not directly provide us with an explicit method of computing \mathbf{A}^{-1} , the covariance matrix associated with $\delta\mathbf{x}^*$, should we desire it. The Cholesky approach is better suited to this end.

Cholesky Decomposition

Every symmetric positive-definite matrix, including \mathbf{A} , can be factored as follows through a Cholesky decomposition:

$$\underbrace{\begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{12}^T & \mathbf{A}_{22} \end{bmatrix}}_{\mathbf{A}} = \underbrace{\begin{bmatrix} \mathbf{U}_{11} & \mathbf{U}_{12} \\ \mathbf{0} & \mathbf{U}_{22} \end{bmatrix}}_{\mathbf{U}} \underbrace{\begin{bmatrix} \mathbf{U}_{11}^T & \mathbf{0} \\ \mathbf{U}_{12}^T & \mathbf{U}_{22}^T \end{bmatrix}}_{\mathbf{U}^T}, \quad (9.32)$$

where \mathbf{U} is an upper-triangular matrix. Multiplying this out reveals

$$\mathbf{U}_{22}\mathbf{U}_{22}^T = \mathbf{A}_{22} : \text{cheap to compute } \mathbf{U}_{22} \text{ via Cholesky due to } \mathbf{A}_{22} \text{ block-diagonal,}$$

$$\mathbf{U}_{12}\mathbf{U}_{22}^T = \mathbf{A}_{12} : \text{cheap to solve for } \mathbf{U}_{12} \text{ due to } \mathbf{U}_{22} \text{ block-diagonal,}$$

$$\mathbf{U}_{11}\mathbf{U}_{11}^T + \mathbf{U}_{12}\mathbf{U}_{12}^T = \mathbf{A}_{11} : \text{cheap to compute } \mathbf{U}_{11} \text{ via Cholesky due to small size of } \delta\mathbf{x}_1^*,$$

so that we have a procedure to very efficiently compute \mathbf{U} , owing to the sparsity of \mathbf{A}_{22} . Note that \mathbf{U}_{22} is also block-diagonal.

If all we cared about was efficiently solving (9.31), then after computing the Cholesky decomposition we can do so in two steps. First, solve

$$\mathbf{U}\mathbf{c} = \mathbf{b}, \quad (9.33)$$

for a temporary variable, \mathbf{c} . This can be done very quickly since \mathbf{U} is upper-triangular and so can be solved from the bottom to the top through substitution and exploiting the additional known sparsity of \mathbf{U} . Second, solve

$$\mathbf{U}^T\delta\mathbf{x}^* = \mathbf{c}, \quad (9.34)$$

for $\delta\mathbf{x}^*$. Again, since \mathbf{U}^T is lower-triangular we can solve quickly from the top to the bottom through substitution and exploiting the sparsity.

Alternatively, we can invert \mathbf{U} directly so that

$$\begin{bmatrix} \mathbf{U}_{11} & \mathbf{U}_{12} \\ \mathbf{0} & \mathbf{U}_{22} \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{U}_{11}^{-1} & -\mathbf{U}_{11}^{-1}\mathbf{U}_{12}\mathbf{U}_{22}^{-1} \\ \mathbf{0} & \mathbf{U}_{22}^{-1} \end{bmatrix}, \quad (9.35)$$

which can again be computed efficiently due to the fact that \mathbf{U}_{22} is block-diagonal and \mathbf{U}_{11} is small and in upper-triangular form. Then we have that

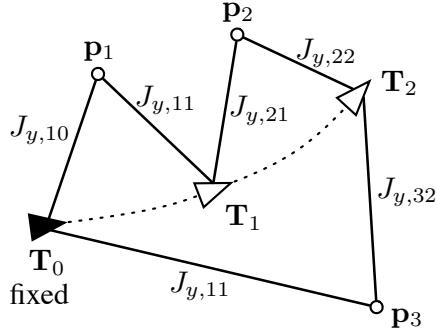
$$\mathbf{U}^T\delta\mathbf{x}^* = \mathbf{U}^{-1}\mathbf{b}, \quad (9.36)$$

or

$$\begin{bmatrix} \mathbf{U}_{11}^T & \mathbf{0} \\ \mathbf{U}_{12}^T & \mathbf{U}_{22}^T \end{bmatrix} \begin{bmatrix} \delta\mathbf{x}_1^* \\ \delta\mathbf{x}_2^* \end{bmatrix} = \begin{bmatrix} \mathbf{U}_{11}^{-1}(\mathbf{b}_1 - \mathbf{U}_{12}\mathbf{U}_{22}^{-1}\mathbf{b}_2) \\ \mathbf{U}_{22}^{-1}\mathbf{b}_2 \end{bmatrix}, \quad (9.37)$$

which allows us to compute $\delta\mathbf{x}_1^*$ and then back-substitute for $\delta\mathbf{x}_2^*$, similarly to the Schur complement method.

Figure 9.3 A BA problem with only three (non-collinear) point landmarks and two free poses (1 and 2). Pose 0 is fixed. It turns out this problem does not have a unique solution as there are too few landmarks to constrain the two free poses. There is one term in the cost function, $J_{y,jk}$, for each measurement, as shown.



However, unlike the Schur complement method, \mathbf{A}^{-1} is now computed easily:

$$\begin{aligned}\mathbf{A}^{-1} &= (\mathbf{U}\mathbf{U}^T)^{-1} = \mathbf{U}^{-T}\mathbf{U}^{-1} = \mathbf{L}\mathbf{L}^T \\ &= \underbrace{\begin{bmatrix} \mathbf{U}_{11}^{-T} & \mathbf{0} \\ -\mathbf{U}_{22}^{-T}\mathbf{U}_{12}^T\mathbf{U}_{11}^{-T} & \mathbf{U}_{22}^{-T} \end{bmatrix}}_{\mathbf{L}} \underbrace{\begin{bmatrix} \mathbf{U}_{11}^{-1} & -\mathbf{U}_{11}^{-1}\mathbf{U}_{12}\mathbf{U}_{22}^{-1} \\ \mathbf{0} & \mathbf{U}_{22}^{-1} \end{bmatrix}}_{\mathbf{L}^T} \\ &= \begin{bmatrix} \mathbf{U}_{11}^{-T}\mathbf{U}_{11}^{-1} & -\mathbf{U}_{11}^{-T}\mathbf{U}_{11}^{-1}\mathbf{U}_{12}\mathbf{U}_{22}^{-1} \\ -\mathbf{U}_{22}^{-T}\mathbf{U}_{12}^T\mathbf{U}_{11}^{-T}\mathbf{U}_{11}^{-1} & \mathbf{U}_{22}^{-T}(\mathbf{U}_{12}^T\mathbf{U}_{11}^{-T}\mathbf{U}_{11}^{-1}\mathbf{U}_{12} + \mathbf{1})\mathbf{U}_{22}^{-1} \end{bmatrix}, \quad (9.38)\end{aligned}$$

where we see additional room for efficiency through repeated products inside the final matrix.

9.1.5 Interpolation Example

It will be instructive to work out the details for the small BA problem in Figure 9.3. There are three (non-collinear) point landmarks and two free poses (1 and 2). We will assume pose 0 is fixed to make the problem observable. We will also assume that the measurements of our point landmarks are three-dimensional; thus the sensor could be either a stereo camera or a range-azimuth-elevation sensor, for example. Unfortunately, there is not enough information present to uniquely solve for the two free poses as well as the positions of the three landmarks. This type of situation arises in rolling-shutter cameras and scanning-while-moving laser sensors.

In the absence of any measurements of additional landmarks, our only recourse is to assume something about the trajectory the vehicle has taken. There are essentially two possibilities:

- (i) Penalty Term: we can take a *maximum a posteriori (MAP)* approach that assumes a prior density over trajectories and encourages the solver to select a likely trajectory that is compatible with the measurements by introducing a penalty term

in the cost function. This is essentially the *simultaneous localization and mapping (SLAM)* approach and will be treated in the next section.

- (ii) Constraint: we can stick with a *maximum likelihood (ML)* approach, but constrain the trajectory to be of a particular form. Here we will do this by assuming the vehicle has a constant six-degree-of-freedom velocity between poses 0 and 2 so that we can use pose interpolation for pose 1. This reduces the number of free pose variables from two to one and provides a unique solution.

We will first set up the equations as though it were possible to solve for both poses 1 and 2 and then introduce the pose-interpolation scheme.

The state variables to be estimated are

$$\mathbf{x} = \{\mathbf{T}_1, \mathbf{T}_2, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3\}. \quad (9.39)$$

We use the usual perturbation schemes,

$$\mathbf{T}_k = \exp(\boldsymbol{\epsilon}_k^\wedge) \mathbf{T}_{\text{op},k}, \quad \mathbf{p}_j = \mathbf{p}_{\text{op},j} + \mathbf{D}\boldsymbol{\zeta}_j, \quad (9.40)$$

and stack our perturbation variables as

$$\delta\mathbf{x} = \begin{bmatrix} \boldsymbol{\epsilon}_1 \\ \boldsymbol{\epsilon}_2 \\ \boldsymbol{\zeta}_1 \\ \boldsymbol{\zeta}_2 \\ \boldsymbol{\zeta}_3 \end{bmatrix}. \quad (9.41)$$

At each iteration, the optimal perturbation variables should be the solution to the linear system,

$$\mathbf{A} \delta\mathbf{x}^* = \mathbf{b}, \quad (9.42)$$

where the \mathbf{A} and \mathbf{b} matrices for this problem have the form

$$\mathbf{A} = \left[\begin{array}{cc|ccc} \mathbf{A}_{11} & \mathbf{A}_{22} & \mathbf{A}_{13} & \mathbf{A}_{14} & \mathbf{A}_{25} \\ \mathbf{A}_{13}^T & \mathbf{A}_{14}^T & \mathbf{A}_{33} & \mathbf{A}_{44} & \mathbf{A}_{55} \\ \hline \mathbf{A}_{14}^T & \mathbf{A}_{24}^T & & & \end{array} \right], \quad \mathbf{b} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \mathbf{b}_3 \\ \mathbf{b}_4 \\ \mathbf{b}_5 \end{bmatrix}, \quad (9.43)$$

with

$$\begin{aligned}\mathbf{A}_{11} &= \mathbf{G}_{1,11}^T \mathbf{R}_{11}^{-1} \mathbf{G}_{1,11} + \mathbf{G}_{1,21}^T \mathbf{R}_{21}^{-1} \mathbf{G}_{1,21}, \\ \mathbf{A}_{22} &= \mathbf{G}_{1,22}^T \mathbf{R}_{22}^{-1} \mathbf{G}_{1,22} + \mathbf{G}_{1,32}^T \mathbf{R}_{32}^{-1} \mathbf{G}_{1,32}, \\ \mathbf{A}_{33} &= \mathbf{G}_{2,10}^T \mathbf{R}_{10}^{-1} \mathbf{G}_{2,10} + \mathbf{G}_{2,11}^T \mathbf{R}_{11}^{-1} \mathbf{G}_{2,11}, \\ \mathbf{A}_{44} &= \mathbf{G}_{2,21}^T \mathbf{R}_{21}^{-1} \mathbf{G}_{2,21} + \mathbf{G}_{2,22}^T \mathbf{R}_{22}^{-1} \mathbf{G}_{2,22}, \\ \mathbf{A}_{55} &= \mathbf{G}_{2,30}^T \mathbf{R}_{30}^{-1} \mathbf{G}_{2,30} + \mathbf{G}_{2,32}^T \mathbf{R}_{32}^{-1} \mathbf{G}_{2,32}, \\ \mathbf{A}_{13} &= \mathbf{G}_{1,11}^T \mathbf{R}_{11}^{-1} \mathbf{G}_{2,11}, \\ \mathbf{A}_{14} &= \mathbf{G}_{1,21}^T \mathbf{R}_{21}^{-1} \mathbf{G}_{2,21}, \\ \mathbf{A}_{24} &= \mathbf{G}_{1,22}^T \mathbf{R}_{22}^{-1} \mathbf{G}_{2,22}, \\ \mathbf{A}_{25} &= \mathbf{G}_{1,32}^T \mathbf{R}_{32}^{-1} \mathbf{G}_{2,32},\end{aligned}$$

and

$$\begin{aligned}\mathbf{b}_1 &= \mathbf{G}_{1,11}^T \mathbf{R}_{11}^{-1} \mathbf{e}_{y,11}(\mathbf{x}_{\text{op}}) + \mathbf{G}_{1,21}^T \mathbf{R}_{21}^{-1} \mathbf{e}_{y,21}(\mathbf{x}_{\text{op}}), \\ \mathbf{b}_2 &= \mathbf{G}_{1,22}^T \mathbf{R}_{22}^{-1} \mathbf{e}_{y,22}(\mathbf{x}_{\text{op}}) + \mathbf{G}_{1,32}^T \mathbf{R}_{32}^{-1} \mathbf{e}_{y,32}(\mathbf{x}_{\text{op}}), \\ \mathbf{b}_3 &= \mathbf{G}_{2,10}^T \mathbf{R}_{10}^{-1} \mathbf{e}_{y,10}(\mathbf{x}_{\text{op}}) + \mathbf{G}_{2,11}^T \mathbf{R}_{11}^{-1} \mathbf{e}_{y,11}(\mathbf{x}_{\text{op}}), \\ \mathbf{b}_4 &= \mathbf{G}_{2,21}^T \mathbf{R}_{21}^{-1} \mathbf{e}_{y,21}(\mathbf{x}_{\text{op}}) + \mathbf{G}_{2,22}^T \mathbf{R}_{22}^{-1} \mathbf{e}_{y,22}(\mathbf{x}_{\text{op}}), \\ \mathbf{b}_5 &= \mathbf{G}_{2,30}^T \mathbf{R}_{30}^{-1} \mathbf{e}_{y,30}(\mathbf{x}_{\text{op}}) + \mathbf{G}_{2,32}^T \mathbf{R}_{32}^{-1} \mathbf{e}_{y,32}(\mathbf{x}_{\text{op}}).\end{aligned}$$

Unfortunately, \mathbf{A} is not invertible in this situation, which means that we cannot solve for $\delta\mathbf{x}^*$ at any iteration.

To remedy the problem, we will assume that the vehicle has followed a constant-velocity trajectory so that we can write \mathbf{T}_1 in terms of \mathbf{T}_2 using the pose-interpolation scheme of Section 7.1.7. To do this, we require the times corresponding to each pose:

$$t_0, t_1, t_2. \quad (9.44)$$

We then define the interpolation variable,

$$\alpha = \frac{t_1 - t_0}{t_2 - t_0}, \quad (9.45)$$

so that we can write

$$\mathbf{T}_1 = \mathbf{T}^\alpha, \quad (9.46)$$

where $\mathbf{T} = \mathbf{T}_2$. Our usual perturbation scheme is

$$\mathbf{T} = \exp(\epsilon^\wedge) \mathbf{T}_{\text{op}} \approx (\mathbf{1} + \epsilon^\wedge) \mathbf{T}_{\text{op}}, \quad (9.47)$$

and for the interpolated variable we have

$$\mathbf{T}^\alpha = (\exp(\epsilon^\wedge) \mathbf{T}_{\text{op}})^\alpha \approx \left(\mathbf{1} + (\mathcal{A}(\alpha, \xi_{\text{op}}) \epsilon)^\wedge \right) \mathbf{T}_{\text{op}}^\alpha, \quad (9.48)$$

where \mathcal{A} is the interpolation Jacobian and $\xi_{\text{op}} = \ln(\mathbf{T}_{\text{op}})^\vee$. Using this

pose-interpolation scheme, we can write the old stacked perturbation variables in terms of a new reduced set:

$$\underbrace{\begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \zeta_1 \\ \zeta_2 \\ \zeta_3 \end{bmatrix}}_{\delta \mathbf{x}} = \underbrace{\begin{bmatrix} \mathcal{A}(\alpha, \xi_{\text{op}}) & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \\ & & & & 1 \end{bmatrix}}_{\mathcal{I}} \underbrace{\begin{bmatrix} \epsilon \\ \zeta_1 \\ \zeta_2 \\ \zeta_3 \end{bmatrix}}_{\delta \mathbf{x}'}, \quad (9.49)$$

where we will call \mathcal{I} the *interpolation matrix*. Our new set of state variables to be estimated is

$$\mathbf{x}' = \{\mathbf{T}, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3\}, \quad (9.50)$$

now that we have eliminated \mathbf{T}_1 as a free variable. Returning to our original ML cost function, we can now rewrite it as

$$J(\mathbf{x}') \approx J(\mathbf{x}'_{\text{op}}) - \mathbf{b}'^T \delta \mathbf{x}' + \frac{1}{2} \delta \mathbf{x}'^T \mathbf{A}' \delta \mathbf{x}', \quad (9.51)$$

where

$$\mathbf{A}' = \mathcal{I}^T \mathbf{A} \mathcal{I}, \quad \mathbf{b}' = \mathcal{I}^T \mathbf{b}. \quad (9.52)$$

The optimal perturbation (that minimizes the cost function), $\delta \mathbf{x}'^*$, is now the solution to

$$\mathbf{A}' \delta \mathbf{x}'^* = \mathbf{b}'. \quad (9.53)$$

We update all the operating points in

$$\mathbf{x}'_{\text{op}} = \{\mathbf{T}_{\text{op}}, \mathbf{p}_{\text{op},1}, \mathbf{p}_{\text{op},2}, \mathbf{p}_{\text{op},3}\}, \quad (9.54)$$

using the usual schemes,

$$\mathbf{T}_{\text{op}} \leftarrow \exp(\epsilon^{*\wedge}) \mathbf{T}_{\text{op}}, \quad \mathbf{p}_{\text{op},j} \leftarrow \mathbf{p}_{\text{op},j} + \mathbf{D} \zeta_j^*, \quad (9.55)$$

and iterate to convergence.

Importantly, applying the interpolation matrix on either side of \mathbf{A} to create \mathbf{A}' does not completely destroy the sparsity. In fact, the bottom-right block corresponding to the landmarks remains block-diagonal, and thus \mathbf{A}' is still an arrowhead matrix:

$$\mathbf{A}' = \left[\begin{array}{c|ccc} * & * & * & * \\ * & * & & \\ * & & * & \\ * & & & * \end{array} \right], \quad (9.56)$$

where $*$ indicates a non-zero block. This means that we can still exploit the sparsity using the methods of the previous section, while interpolating poses.

It turns out that we can use this interpolation scheme (and others) for more complicated BA problems as well. We just need to decide which

pose variables we want to keep in the state and which to interpolate, then build the appropriate interpolation matrix, \mathcal{I} .

9.2 Simultaneous Localization and Mapping

The SLAM problem is essentially the same as BA, except that we also typically know something about how the vehicle has moved (i.e., a motion model) and can therefore include inputs, \mathbf{v} , in the problem. Logistically, we only need to augment the BA cost function with additional terms corresponding to the inputs (Sibley, 2006). Smith et al. (1990) is the classic reference on SLAM and Durrant-Whyte and Bailey (2006); Bailey and Durrant-Whyte (2006) provide a detailed survey of the area. The difference is essentially that BA is a *maximum likelihood (ML)* problem and SLAM is a *maximum a posteriori (MAP)* problem. Our approach is a batch-SLAM method (Lu and Milios, 1997) similar to the Graph SLAM approach of Thrun and Montemerlo (2005), but using our method of handling pose variables in three-dimensional space.

9.2.1 Problem Setup

Another minor difference is that by including inputs/priors, we can also assume that we have a prior on the initial state, \mathbf{T}_0 , so that it can be included in the estimation problem (unlike BA)⁴. Our state to be estimated is thus

$$\mathbf{x} = \{\mathbf{T}_0, \dots, \mathbf{T}_K, \mathbf{p}_1, \dots, \mathbf{p}_M\}. \quad (9.57)$$

We assume the same measurement model as the BA problem, and the measurements are given by

$$\mathbf{y} = \{\mathbf{y}_{10}, \dots, \mathbf{y}_{M0}, \dots, \mathbf{y}_{1K}, \dots, \mathbf{y}_{MK}\}. \quad (9.58)$$

We will adopt the motion model from Section 8.2 and the inputs are given by

$$\mathbf{v} = \{\check{\mathbf{T}}_0, \boldsymbol{\varpi}_1, \boldsymbol{\varpi}_2, \dots, \boldsymbol{\varpi}_K\}. \quad (9.59)$$

We will next set up the batch MAP problem.

⁴ We could also choose not to estimate it and simply hold it fixed, which is very common.

9.2.2 Batch Maximum a Posteriori Solution

We define the following matrices:

$$\begin{aligned}\delta \mathbf{x} &= \begin{bmatrix} \delta \mathbf{x}_1 \\ \delta \mathbf{x}_2 \end{bmatrix}, \quad \mathbf{H} = \begin{bmatrix} \mathbf{F}^{-1} & \mathbf{0} \\ \mathbf{G}_1 & \mathbf{G}_2 \end{bmatrix}, \quad \mathbf{W} = \begin{bmatrix} \mathbf{Q} & \mathbf{0} \\ \mathbf{0} & \mathbf{R} \end{bmatrix}, \\ \mathbf{e}(\mathbf{x}_{\text{op}}) &= \begin{bmatrix} \mathbf{e}_v(\mathbf{x}_{\text{op}}) \\ \mathbf{e}_y(\mathbf{x}_{\text{op}}) \end{bmatrix},\end{aligned}\quad (9.60)$$

where

$$\begin{aligned}\delta \mathbf{x}_1 &= \begin{bmatrix} \epsilon_0 \\ \epsilon_1 \\ \vdots \\ \epsilon_K \end{bmatrix}, \quad \delta \mathbf{x}_2 = \begin{bmatrix} \zeta_1 \\ \zeta_2 \\ \vdots \\ \zeta_M \end{bmatrix}, \\ \mathbf{e}_v(\mathbf{x}_{\text{op}}) &= \begin{bmatrix} \mathbf{e}_{v,0}(\mathbf{x}_{\text{op}}) \\ \mathbf{e}_{v,1}(\mathbf{x}_{\text{op}}) \\ \vdots \\ \mathbf{e}_{v,K}(\mathbf{x}_{\text{op}}) \end{bmatrix}, \quad \mathbf{e}_y(\mathbf{x}_{\text{op}}) = \begin{bmatrix} \mathbf{e}_{y,10}(\mathbf{x}_{\text{op}}) \\ \mathbf{e}_{y,20}(\mathbf{x}_{\text{op}}) \\ \vdots \\ \mathbf{e}_{y,MK}(\mathbf{x}_{\text{op}}) \end{bmatrix}, \\ \mathbf{Q} &= \text{diag}(\check{\mathbf{P}}_0, \mathbf{Q}_1, \dots, \mathbf{Q}_K), \quad \mathbf{R} = \text{diag}(\mathbf{R}_{10}, \mathbf{R}_{20}, \dots, \mathbf{R}_{MK}), \\ \mathbf{F}^{-1} &= \begin{bmatrix} \mathbf{1} & & & & \\ -\mathbf{F}_0 & \mathbf{1} & & & \\ & -\mathbf{F}_1 & \ddots & & \\ & & \ddots & \mathbf{1} & \\ & & & -\mathbf{F}_{K-1} & \mathbf{1} \end{bmatrix}, \quad (9.61) \\ \mathbf{G}_1 &= \begin{bmatrix} \mathbf{G}_{1,10} & & & & \\ \vdots & & & & \\ \mathbf{G}_{1,M0} & \mathbf{G}_{1,11} & & & \\ & \vdots & & & \\ & \mathbf{G}_{1,M1} & & & \\ & & \ddots & & \\ & & & \mathbf{G}_{1,1K} & \\ & & & \vdots & \\ & & & \mathbf{G}_{1,MK} & \end{bmatrix}, \quad \mathbf{G}_2 = \begin{bmatrix} \mathbf{G}_{2,10} & & & & \\ \ddots & & & & \\ \mathbf{G}_{2,M0} & & & & \\ & \ddots & & & \\ & & \mathbf{G}_{2,11} & & \\ & & & \ddots & \\ & & & & \mathbf{G}_{2,M1} \\ & & & & \vdots \\ & & & & \mathbf{G}_{2,1K} \\ & & & & \vdots \\ & & & & \mathbf{G}_{2,MK} \end{bmatrix}.\end{aligned}$$

From Sections 8.2.5 for the motion priors and 9.1.3 for the measurements, the detailed blocks are

$$\begin{aligned}\mathbf{F}_{k-1} &= \text{Ad}(\mathbf{T}_{\text{op},k} \mathbf{T}_{\text{op},k-1}^{-1}), \quad k = 1 \dots K, \\ \mathbf{e}_{v,k}(\mathbf{x}_{\text{op}}) &= \begin{cases} \ln(\check{\mathbf{T}}_0 \mathbf{T}_{\text{op},0}^{-1})^\vee & k = 0 \\ \ln(\exp((t_k - t_{k-1}) \boldsymbol{\varpi}_k^\wedge) \mathbf{T}_{\text{op},k-1} \mathbf{T}_{\text{op},k}^{-1})^\vee & k = 1 \dots K \end{cases}, \\ \mathbf{G}_{1,jk} &= \mathbf{S}_{jk} (\mathbf{T}_{\text{op},k} \mathbf{p}_{\text{op},j})^\odot, \quad \mathbf{G}_{2,jk} = \mathbf{S}_{jk} \mathbf{T}_{\text{op},k} \mathbf{D}, \\ \mathbf{e}_{y,jk}(\mathbf{x}_{\text{op}}) &= \mathbf{y}_{jk} - \mathbf{s}(\mathbf{T}_{\text{op},k} \mathbf{p}_{\text{op},j}).\end{aligned}\quad (9.62)$$

Finally, the objective function can be written as usual as

$$J(\mathbf{x}) \approx J(\mathbf{x}_{\text{op}}) - \mathbf{b}^T \delta \mathbf{x} + \frac{1}{2} \delta \mathbf{x}^T \mathbf{A} \delta \mathbf{x}, \quad (9.63)$$

where

$$\mathbf{A} = \mathbf{H}^T \mathbf{W}^{-1} \mathbf{H}, \quad \mathbf{b} = \mathbf{H}^T \mathbf{W}^{-1} \mathbf{e}(\mathbf{x}_{\text{op}}), \quad (9.64)$$

whereupon the minimizing perturbations, $\delta \mathbf{x}^*$, are the solutions to

$$\mathbf{A} \delta \mathbf{x}^* = \mathbf{b}. \quad (9.65)$$

We solve for $\delta \mathbf{x}^*$, then update our operating points according to

$$\mathbf{T}_{\text{op},k} \leftarrow \exp(\boldsymbol{\epsilon}_k^{*\wedge}) \mathbf{T}_{\text{op},k}, \quad \mathbf{p}_{\text{op},j} \leftarrow \mathbf{p}_{\text{op},j} + \mathbf{D} \boldsymbol{\zeta}_j^*, \quad (9.66)$$

and iterate to convergence. As in the BA case, once converged we set $\hat{\mathbf{T}}_{v_k i} = \mathbf{T}_{\text{op},k}$ and $\hat{\mathbf{p}}_i^{p_j i} = \mathbf{p}_{\text{op},j}$ at the last iteration as the final estimates for the vehicle poses and landmark positions of interest.

9.2.3 Exploiting Sparsity

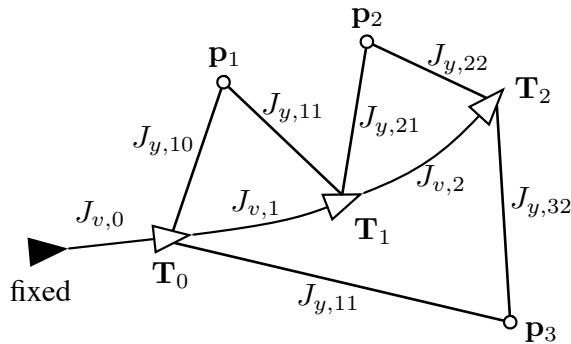
Introducing the motion priors does not destroy the nice sparsity of the original BA problem. We can see this by noting that

$$\begin{aligned}\mathbf{A} &= \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{12}^T & \mathbf{A}_{22} \end{bmatrix} = \mathbf{H}^T \mathbf{W}^{-1} \mathbf{H} \\ &= \begin{bmatrix} \mathbf{F}^{-T} \mathbf{Q}^{-1} \mathbf{F}^{-1} + \mathbf{G}_1^T \mathbf{R}^{-1} \mathbf{G}_1 & \mathbf{G}_1^T \mathbf{R}^{-1} \mathbf{G}_2 \\ \mathbf{G}_2^T \mathbf{R}^{-1} \mathbf{G}_1 & \mathbf{G}_2^T \mathbf{R}^{-1} \mathbf{G}_2 \end{bmatrix}.\end{aligned}\quad (9.67)$$

Compared to the BA problem, blocks \mathbf{A}_{12} and \mathbf{A}_{22} have not changed at all, showing that \mathbf{A} is still an arrowhead matrix with \mathbf{A}_{22} block-diagonal. We can thus exploit this sparsity to solve for the perturbations at each iteration efficiently using either the Schur or Cholesky methods.

While block \mathbf{A}_{11} is now different than the BA problem,

$$\mathbf{A}_{11} = \underbrace{\mathbf{F}^{-T} \mathbf{Q}^{-1} \mathbf{F}^{-1}}_{\text{prior}} + \underbrace{\mathbf{G}_1^T \mathbf{R}^{-1} \mathbf{G}_1}_{\text{measurements}}, \quad (9.68)$$



we have seen previously (e.g., Section 8.2.5) that it is block-tridiagonal. Thus, we could choose to exploit the sparsity of \mathbf{A}_{11} rather than \mathbf{A}_{22} , if the number of poses were large compared to the number of landmarks, for example. In this case, the Cholesky method is preferred over the Schur one as we do not need to construct \mathbf{A}_{11}^{-1} , which is actually dense. Kaess et al. (2008, 2012) provide incremental methods of updating the batch-SLAM solution that exploit sparsity beyond the primary block structure discussed here.

9.2.4 Example

Figure 9.4 shows a simple SLAM problem with three point landmarks and three free poses. In contrast to the BA example of Figure 9.3, we now allow \mathbf{T}_0 to be estimated as we have some prior information about it, $\{\check{\mathbf{T}}_0, \check{\mathbf{P}}_0\}$, in relation to an external reference frame (shown as the black, fixed pose). We have shown graphically all of the terms in the objective function⁵, one for each measurement and input, totalling nine terms:

$$J = \underbrace{J_{v,0} + J_{v,1} + J_{v,2}}_{\text{prior terms}} + \underbrace{J_{y,10} + J_{y,30} + J_{y,11} + J_{y,21} + J_{y,22} + J_{y,32}}_{\text{measurement terms}} \quad (9.69)$$

Also, with the motion priors we have used, \mathbf{A} is always well conditioned and will provide a solution for the trajectory, even without any measurements.

Figure 9.4 A SLAM problem with only three (non-collinear) point landmarks and three free poses. Pose 0 is not fixed as we have some prior information about it. There is one term in the cost function for each measurement, $J_{y,jk}$, and motion prior, $J_{v,k}$, as shown.

⁵ Sometimes this type of diagram is called a *factor graph* with each ‘factor’ from the posterior likelihood over the states becoming a ‘term’ in the objective function, which is really just the negative log likelihood of the posterior over states.

10

Continuous-Time Estimation

All of our examples in this last part of the book have been in discrete time, which is sufficient for many applications. However, it is worth investigating how we might make use of the continuous-time estimation tools from Sections 3.4 and 4.4 when working with state variables in $SE(3)$. To this end, we show one way to start from a specific nonlinear, stochastic differential equation and build motion priors that encourage trajectory smoothness¹. We then show where these motion priors could be used within a trajectory estimation problem. Finally, we show how to interpolate for query poses at times in between the main solution times, using Gaussian process interpolation.

10.1 Motion Prior

We will begin by discussing how to represent general motion priors on $SE(3)$. We will do this in the context of a specific nonlinear, stochastic differential equation. We will also further simplify this to make the analysis tractable.

10.1.1 General

Ideally, we would like to use the following system of nonlinear, stochastic, differential equations to build our motion prior²:

$$\dot{\mathbf{T}}(t) = \boldsymbol{\varpi}(t)^{\wedge} \mathbf{T}(t), \quad (10.1a)$$

$$\dot{\boldsymbol{\varpi}}(t) = \mathbf{w}(t), \quad (10.1b)$$

$$\mathbf{w}(t) \sim \mathcal{GP}(\mathbf{0}, \mathbf{Q} \delta(t - t')). \quad (10.1c)$$

To use this to build our motion priors, we will need to estimate the pose, $\mathbf{T}(t)$, and the body-centric, generalized velocity, $\boldsymbol{\varpi}(t)$, at some times of interest: t_0, t_1, \dots, t_K . White noise, $\mathbf{w}(t)$, enters the system through the generalized angular acceleration; in the absence of noise, the body-centric, generalized velocity is constant. The trouble with using this

¹ See Furgale et al. (2015) for a survey of continuous-time methods.

² It is this model that we approximated as a discrete-time system in the previous two chapters in order to build discrete-time motion priors.

model directly in continuous time is that it is nonlinear and the state is of the form $\{\mathbf{T}(t), \boldsymbol{\varpi}(t)\} \in SE(3) \times \mathbb{R}^3$. Even the nonlinear tools from Section 4.4 do not directly apply in this situation.

Gaussian Processes for $SE(3)$

Inspired by the way we have handled Gaussian random variables for Lie groups, we can define a Gaussian process for poses in which the mean function is defined on $SE(3) \times \mathbb{R}^3$ and the covariance function is defined on $\mathfrak{se}(3) \times \mathbb{R}^3$:

$$\text{mean function: } \{\check{\mathbf{T}}(t), \check{\boldsymbol{\varpi}}\}, \quad (10.2a)$$

$$\text{covariance function: } \check{\mathbf{P}}(t, t'), \quad (10.2b)$$

$$\text{combination: } \mathbf{T}(t) = \exp(\boldsymbol{\xi}(t)^\wedge) \check{\mathbf{T}}(t), \quad (10.2c)$$

$$\boldsymbol{\varpi}(t) = \check{\boldsymbol{\varpi}} + \boldsymbol{\eta}(t), \quad (10.2d)$$

$$\underbrace{\begin{bmatrix} \boldsymbol{\xi}(t) \\ \boldsymbol{\eta}(t) \end{bmatrix}}_{\boldsymbol{\gamma}(t)} \sim \mathcal{GP}(\mathbf{0}, \check{\mathbf{P}}(t, t')). \quad (10.2e)$$

While we could attempt to specify the covariance function directly, we prefer to define it via a stochastic differential equation. Fortunately, we can use (7.253) from Section 7.2.4 to break the above SDE into nominal (mean) and perturbed (noise) parts:

$$\text{nominal (mean): } \dot{\check{\mathbf{T}}}(t) = \check{\boldsymbol{\varpi}}^\wedge \check{\mathbf{T}}(t), \quad (10.3a)$$

$$\dot{\check{\boldsymbol{\varpi}}} = \mathbf{0}, \quad (10.3b)$$

$$\text{perturbation (cov): } \underbrace{\begin{bmatrix} \dot{\boldsymbol{\xi}}(t) \\ \dot{\boldsymbol{\eta}}(t) \end{bmatrix}}_{\dot{\boldsymbol{\gamma}}(t)} = \underbrace{\begin{bmatrix} \check{\boldsymbol{\varpi}}^\wedge & \mathbf{1} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}}_A \underbrace{\begin{bmatrix} \boldsymbol{\xi}(t) \\ \boldsymbol{\eta}(t) \end{bmatrix}}_{\boldsymbol{\gamma}(t)} + \underbrace{\begin{bmatrix} \mathbf{0} \\ \mathbf{1} \end{bmatrix}}_L \mathbf{w}(t), \quad (10.3c)$$

$$\mathbf{w}(t) \sim \mathcal{GP}(\mathbf{0}, \mathbf{Q} \delta(t - t')). \quad (10.3d)$$

The (deterministic) differential equation defining the mean function is nonlinear and can be integrated in closed form,

$$\check{\mathbf{T}}(t) = \exp((t - t_0) \check{\boldsymbol{\varpi}}^\wedge) \check{\mathbf{T}}_0, \quad (10.4)$$

while the SDE defining the covariance function is linear, time-invariant³ and thus we can apply the tools from Section 3.4, specifically 3.4.3, to build a motion prior. It is important to note that this new system of equations only approximates the (ideal) one at the start of the section; they will be very similar when $\boldsymbol{\xi}(t)$ remains small. Anderson and Barfoot (2015) provide a different method of approximating the desired nonlinear SDE that employs local variables.

³ While we have that the prior generalized velocity does not change over time, in general we could use whatever time-varying function we like, $\check{\boldsymbol{\varpi}}(t)$, but the computation of the transition function will become more complicated. Letting it be piecewise constant between measurement times would be a straightforward extension.

Transition Function

With $\check{\boldsymbol{\varpi}}$ constant, the transition function for the perturbation system is

$$\Phi(t, s) = \begin{bmatrix} \exp((t-s)\check{\boldsymbol{\varpi}}^\lambda) & (t-s)\mathcal{J}((t-s)\check{\boldsymbol{\varpi}}) \\ \mathbf{0} & \mathbf{1} \end{bmatrix}, \quad (10.5)$$

with no approximation. This can be checked by verifying that $\Phi(t, t) = \mathbf{1}$ and $\dot{\Phi}(t, s) = \mathbf{A}\Phi(t, s)$. We can also see this by working out the transition function directly for this LTI system:

$$\begin{aligned} \Phi(t, s) &= \exp(\mathbf{A}(t-s)) \\ &= \sum_{n=0}^{\infty} \frac{(t-s)^n}{n!} \mathbf{A}^n \\ &= \sum_{n=0}^{\infty} \frac{(t-s)^n}{n!} \left(\begin{bmatrix} \check{\boldsymbol{\varpi}}^\lambda & \mathbf{1} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \right)^n \\ &= \begin{bmatrix} \sum_{n=0}^{\infty} \frac{(t-s)^n}{n!} (\check{\boldsymbol{\varpi}}^\lambda)^n & (t-s) \sum_{n=0}^{\infty} \frac{(t-s)^n}{(n+1)!} (\check{\boldsymbol{\varpi}}^\lambda)^n \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \\ &= \begin{bmatrix} \exp((t-s)\check{\boldsymbol{\varpi}}^\lambda) & (t-s)\mathcal{J}((t-s)\check{\boldsymbol{\varpi}}) \\ \mathbf{0} & \mathbf{1} \end{bmatrix}. \end{aligned} \quad (10.6)$$

Notably, it is somewhat rare that we can find the transition function so neatly when \mathbf{A} is not nilpotent.

Error Terms

With the transition function in hand, we can define error terms for use within a maximum-a-posterior estimator. The error term at the first timestep will be

$$\mathbf{e}_{v,0}(\mathbf{x}) = -\boldsymbol{\gamma}(t_0) = -\begin{bmatrix} \ln(\mathbf{T}_0 \check{\mathbf{T}}_0^{-1})^\vee \\ \boldsymbol{\varpi}_0 - \check{\boldsymbol{\varpi}} \end{bmatrix}, \quad (10.7)$$

where $\check{\mathbf{P}}_0$ is our initial state uncertainty. For later times, $k = 1 \dots K$, we define the error term to be

$$\mathbf{e}_{v,k}(\mathbf{x}) = \Phi(t_k, t_{k-1})\boldsymbol{\gamma}(t_{k-1}) - \boldsymbol{\gamma}(t_k), \quad (10.8)$$

which takes a bit of explanation. The idea is to formulate the error on $\mathfrak{se}(3) \times \mathbb{R}^3$ since this is where the covariance function lives. It is worth noting that the mean function for $\boldsymbol{\gamma}(t)$ was defined to be zero, making this error definition straightforward. We can also write this as

$$\mathbf{e}_{v,k}(\mathbf{x}) = \Phi(t_k, t_{k-1}) \begin{bmatrix} \ln(\mathbf{T}_{k-1} \check{\mathbf{T}}_{k-1}^{-1})^\vee \\ \boldsymbol{\varpi}_{k-1} - \check{\boldsymbol{\varpi}} \end{bmatrix} - \begin{bmatrix} \ln(\mathbf{T}_k \check{\mathbf{T}}_k^{-1})^\vee \\ \boldsymbol{\varpi}_k - \check{\boldsymbol{\varpi}} \end{bmatrix}, \quad (10.9)$$

in terms of the quantities to be estimated: \mathbf{T}_k , $\boldsymbol{\varpi}_k$, \mathbf{T}_{k-1} , and $\boldsymbol{\varpi}_{k-1}$. The covariance matrix associated with this error is given by

$$\mathbf{Q}_k = \int_{t_{k-1}}^{t_k} \Phi(t_k, s) \mathbf{L} \mathbf{Q} \mathbf{L}^T \Phi(t_k, s)^T ds, \quad (10.10)$$

which is described in more detail in Section 3.4.3; again, we are using the SDE defined on $\mathfrak{se}(3) \times \mathbb{R}^3$ to compute this covariance. Despite having the transition matrix in closed form, this integral is best computed either numerically or by making an approximation or simplification (which we do in the next section). The combined objective function for the entire motion prior is thus given by

$$J_v(\mathbf{x}) = \frac{1}{2} \mathbf{e}_{v,0}(\mathbf{x})^T \check{\mathbf{P}}_0^{-1} \mathbf{e}_{v,0}(\mathbf{x}) + \frac{1}{2} \sum_{k=1}^K \mathbf{e}_{v,k}(\mathbf{x})^T \mathbf{Q}_k^{-1} \mathbf{e}_{v,k}(\mathbf{x}), \quad (10.11)$$

which does not contain any terms associated with the measurements.

Linearized Error Terms

To linearize our error terms defined above, we adopt the $SE(3)$ perturbation scheme for poses and the usual one for generalized velocities,

$$\mathbf{T}_k = \exp(\boldsymbol{\epsilon}_k^\wedge) \mathbf{T}_{\text{op},k}, \quad (10.12a)$$

$$\boldsymbol{\varpi}_k = \boldsymbol{\varpi}_{\text{op},k} + \boldsymbol{\psi}_k, \quad (10.12b)$$

where $\{\mathbf{T}_{\text{op},k}, \boldsymbol{\varpi}_{\text{op},k}\}$ is the operating point and $(\boldsymbol{\epsilon}_k, \boldsymbol{\psi}_k)$ is the perturbation. Using the pose perturbation scheme we see that

$$\begin{aligned} \boldsymbol{\xi}_k &= \ln(\mathbf{T}_k \check{\mathbf{T}}_k^{-1})^\vee = \ln(\exp(\boldsymbol{\epsilon}_k^\wedge) \mathbf{T}_{\text{op},k} \check{\mathbf{T}}_k^{-1})^\vee \\ &= \ln(\exp(\boldsymbol{\epsilon}_k^\wedge) \exp(\boldsymbol{\xi}_{\text{op},k}^\wedge))^\vee \approx \boldsymbol{\xi}_{\text{op},k} + \boldsymbol{\epsilon}_k, \end{aligned} \quad (10.13)$$

where $\boldsymbol{\xi}_{\text{op},k} = \ln(\mathbf{T}_{\text{op},k} \check{\mathbf{T}}_k^{-1})^\vee$. We have used a very approximate version of the BCH formula here, which is only valid if both $\boldsymbol{\epsilon}_k$ and $\boldsymbol{\xi}_{\text{op},k}$ are both quite small. The former is reasonable since $\boldsymbol{\epsilon}_k \rightarrow \mathbf{0}$ as our estimation scheme converges. The latter will be so if the motion prior has low uncertainty; we have essentially already made this assumption in separating our SDE into the nominal and perturbation parts. Inserting this linearization results in the following linearized error terms:

$$\mathbf{e}_{v,k}(\mathbf{x}) \approx \begin{cases} \mathbf{e}_{v,0}(\mathbf{x}_{\text{op}}) - \boldsymbol{\theta}_0 & k = 0 \\ \mathbf{e}_{v,k}(\mathbf{x}_{\text{op}}) + \mathbf{F}_{k-1} \boldsymbol{\theta}_{k-1} - \boldsymbol{\theta}_k & k = 1 \dots K \end{cases}, \quad (10.14)$$

where

$$\boldsymbol{\theta}_k = \begin{bmatrix} \boldsymbol{\epsilon}_k \\ \boldsymbol{\psi}_k \end{bmatrix}, \quad (10.15)$$

is the stacked perturbation for both the pose and generalized velocity at time k and

$$\mathbf{F}_{k-1} = \Phi(t_k, t_{k-1}). \quad (10.16)$$

Defining

$$\begin{aligned} \delta\mathbf{x}_1 &= \begin{bmatrix} \boldsymbol{\theta}_0 \\ \boldsymbol{\theta}_1 \\ \vdots \\ \boldsymbol{\theta}_K \end{bmatrix}, \quad \mathbf{e}_v(\mathbf{x}_{\text{op}}) = \begin{bmatrix} \mathbf{e}_{v,0}(\mathbf{x}_{\text{op}}) \\ \mathbf{e}_{v,1}(\mathbf{x}_{\text{op}}) \\ \vdots \\ \mathbf{e}_{v,K}(\mathbf{x}_{\text{op}}) \end{bmatrix}, \\ \mathbf{F}^{-1} &= \begin{bmatrix} \mathbf{1} & & & \\ -\mathbf{F}_0 & \mathbf{1} & & \\ & -\mathbf{F}_1 & \ddots & \\ & & \ddots & \mathbf{1} \\ & & & -\mathbf{F}_{K-1} & \mathbf{1} \end{bmatrix}, \\ \mathbf{Q} &= \text{diag}(\check{\mathbf{P}}_0, \mathbf{Q}_1, \dots, \mathbf{Q}_K), \end{aligned} \quad (10.17)$$

we can write the approximate motion-prior part of the objective function in block form as

$$J_v(\mathbf{x}) \approx \frac{1}{2} (\mathbf{e}_v(\mathbf{x}_{\text{op}}) - \mathbf{F}^{-1} \delta\mathbf{x}_1)^T \mathbf{Q}^{-1} (\mathbf{e}_v(\mathbf{x}_{\text{op}}) - \mathbf{F}^{-1} \delta\mathbf{x}_1), \quad (10.18)$$

which is quadratic in the perturbation, $\delta\mathbf{x}_1$.

10.1.2 Simplification

Computing the \mathbf{Q}_k blocks in the general case can be done numerically. However, we can evaluate them in closed form fairly easily for the specific case of no rotational motion (in the mean of the prior only). To do this, we define the (constant) generalized velocity to be of the form

$$\check{\boldsymbol{\omega}} = \begin{bmatrix} \check{\boldsymbol{\nu}} \\ \mathbf{0} \end{bmatrix}. \quad (10.19)$$

The mean function will be a constant, linear velocity (i.e., no angular rotation), $\check{\boldsymbol{\nu}}$. We then have that

$$\check{\boldsymbol{\omega}}^\wedge \check{\boldsymbol{\omega}}^\wedge = \begin{bmatrix} \mathbf{0} & \check{\boldsymbol{\nu}}^\wedge \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{0} & \check{\boldsymbol{\nu}}^\wedge \\ \mathbf{0} & \mathbf{0} \end{bmatrix} = \mathbf{0}, \quad (10.20)$$

so that

$$\exp((t-s)\check{\boldsymbol{\omega}}^\wedge) = \mathbf{1} + (t-s)\check{\boldsymbol{\omega}}^\wedge, \quad (10.21a)$$

$$\mathcal{J}((t-s)\check{\boldsymbol{\omega}}) = \mathbf{1} + \frac{1}{2}(t-s)\check{\boldsymbol{\omega}}^\wedge, \quad (10.21b)$$

with no approximation. The transition function is therefore

$$\Phi(t, s) = \begin{bmatrix} \mathbf{1} + (t-s)\check{\varpi}^\lambda & (t-s)\mathbf{1} + \frac{1}{2}(t-s)^2\check{\varpi}^\lambda \\ \mathbf{0} & \mathbf{1} \end{bmatrix}. \quad (10.22)$$

We now turn to computing the \mathbf{Q}_k blocks starting from

$$\mathbf{Q}_k = \int_{t_{k-1}}^{t_k} \Phi(t_k, s) \mathbf{L} \mathbf{Q} \mathbf{L}^T \Phi(t_k, s)^T ds. \quad (10.23)$$

Plugging in the quantities involved in the integrand we have

$$\mathbf{Q}_k = \begin{bmatrix} \mathbf{Q}_{k,11} & \mathbf{Q}_{k,12} \\ \mathbf{Q}_{k,12}^T & \mathbf{Q}_{k,22} \end{bmatrix}, \quad (10.24a)$$

$$\begin{aligned} \mathbf{Q}_{k,11} = \int_0^{\Delta t_{k:k-1}} & \left((\Delta t_{k:k-1} - s)^2 \mathbf{Q} + \frac{1}{2}(\Delta t_{k:k-1} - s)^3 \right. \\ & \times \left. (\check{\varpi}^\lambda \mathbf{Q} + \mathbf{Q} \check{\varpi}^{\lambda T}) + \frac{1}{4}(\Delta t_{k:k-1} - s)^4 \check{\varpi}^\lambda \mathbf{Q} \check{\varpi}^{\lambda T} \right) ds, \end{aligned} \quad (10.24b)$$

$$\mathbf{Q}_{k,12} = \int_0^{\Delta t_{k:k-1}} \left((\Delta t_{k:k-1} - s) \mathbf{Q} + \frac{1}{2}(\Delta t_{k:k-1} - s)^2 \check{\varpi}^\lambda \mathbf{Q} \right) ds, \quad (10.24c)$$

$$\mathbf{Q}_{k,22} = \int_0^{\Delta t_{k:k-1}} \mathbf{Q} ds, \quad (10.24d)$$

$$\Delta t_{k:k-1} = t_k - t_{k-1}. \quad (10.24e)$$

Carrying out the integrals (of simple polynomials) we have

$$\begin{aligned} \mathbf{Q}_{k,11} = \frac{1}{3} \Delta t_{k:k-1}^3 \mathbf{Q} + \frac{1}{8} \Delta t_{k:k-1}^4 & \left(\check{\varpi}^\lambda \mathbf{Q} + \mathbf{Q} \check{\varpi}^{\lambda T} \right) \\ & + \frac{1}{20} \Delta t_{k:k-1}^5 \check{\varpi}^\lambda \mathbf{Q} \check{\varpi}^{\lambda T}, \end{aligned} \quad (10.25a)$$

$$\mathbf{Q}_{k,12} = \frac{1}{2} \Delta t_{k:k-1}^2 \mathbf{Q} + \frac{1}{6} \Delta t_{k:k-1}^3 \check{\varpi}^\lambda \mathbf{Q}, \quad (10.25b)$$

$$\mathbf{Q}_{k,22} = \Delta t_{k:k-1} \mathbf{Q}. \quad (10.25c)$$

These expressions can be used to build \mathbf{Q}_k from the known quantities, $\check{\varpi}$, \mathbf{Q} , and $\Delta t_{k:k-1}$.

10.2 Simultaneous Trajectory Estimation and Mapping

Using the motion prior from the previous section to smooth the solution, we can set up a *simultaneous trajectory estimation and mapping (STEAM)* problem. STEAM is really just a variant of *simultaneous localization and mapping (SLAM)*, where we have the ability to inherently query the robot's underlying continuous-time trajectory at any time of interest, not just the measurement times. We show first

how to solve for the state at the measurement times. Then, we show how to use Gaussian process interpolation to solve for the state (and covariance) at other query times.

10.2.1 Problem Setup

The use of our continuous-time motion prior is a fairly straightforward modification of the discrete-time approach from Section 9.2.2. The state to be estimated is now

$$\mathbf{x} = \{\mathbf{T}_0, \boldsymbol{\varpi}_0, \dots, \mathbf{T}_K, \boldsymbol{\varpi}_K, \mathbf{p}_1, \dots, \mathbf{p}_M\}, \quad (10.26)$$

which includes the poses, the generalized velocity variables, and the landmark positions. The times, t_0, t_1, \dots, t_K correspond to the measurements times and the measurements available in our problem are

$$\mathbf{y} = \{\mathbf{y}_{10}, \dots, \mathbf{y}_{M0}, \dots, \mathbf{y}_{1K}, \dots, \mathbf{y}_{MK}\}, \quad (10.27)$$

which remain the same as the discrete-time SLAM case.

10.2.2 Measurement Model

We use the same measurement model as the discrete-time SLAM case but slightly modify some of the block matrices to account for the fact that the estimated state now includes the $\boldsymbol{\varpi}_k$ quantities, which are not required in the measurement error terms. We continue to use the usual perturbation scheme for the landmark positions:

$$\mathbf{p}_j = \mathbf{p}_{\text{op},j} + \mathbf{D}\boldsymbol{\zeta}_j, \quad (10.28)$$

where $\mathbf{p}_{\text{op},j}$ is the operating point and $\boldsymbol{\zeta}_j$ is the perturbation.

To build the part of the objective function associated with the measurements, we define the following matrices:

$$\begin{aligned} \delta \mathbf{x}_2 &= \begin{bmatrix} \boldsymbol{\zeta}_1 \\ \boldsymbol{\zeta}_2 \\ \vdots \\ \boldsymbol{\zeta}_M \end{bmatrix}, \quad \mathbf{e}_y(\mathbf{x}_{\text{op}}) = \begin{bmatrix} \mathbf{e}_{y,10}(\mathbf{x}_{\text{op}}) \\ \mathbf{e}_{y,20}(\mathbf{x}_{\text{op}}) \\ \vdots \\ \mathbf{e}_{y,MK}(\mathbf{x}_{\text{op}}) \end{bmatrix}, \\ \mathbf{R} &= \text{diag}(\mathbf{R}_{10}, \mathbf{R}_{20}, \dots, \mathbf{R}_{MK}), \end{aligned} \quad (10.29)$$

and

$$\mathbf{G}_1 = \begin{bmatrix} \mathbf{G}_{1,10} & & & \\ \vdots & \ddots & & \\ \mathbf{G}_{1,M0} & & \mathbf{G}_{1,11} & \\ & & \vdots & \\ & & \mathbf{G}_{1,M1} & \\ & & & \ddots & \\ & & & & \mathbf{G}_{1,1K} \\ & & & & \vdots \\ & & & & \mathbf{G}_{1,MK} \end{bmatrix}, \quad \mathbf{G}_2 = \begin{bmatrix} \mathbf{G}_{2,10} & & & \\ \vdots & \ddots & & \\ \mathbf{G}_{2,M0} & & \mathbf{G}_{2,11} & \\ & & \vdots & \\ & & \mathbf{G}_{2,M1} & \\ & & & \vdots \\ & & & & \mathbf{G}_{2,1K} \\ & & & & \vdots \\ & & & & \mathbf{G}_{2,MK} \end{bmatrix}. \quad (10.30)$$

The detailed blocks are

$$\begin{aligned} \mathbf{G}_{1,jk} &= [\mathbf{S}_{jk} (\mathbf{T}_{\text{op},k} \mathbf{p}_{\text{op},j})^\odot \quad \mathbf{0}], \quad \mathbf{G}_{2,jk} = \mathbf{S}_{jk} \mathbf{T}_{\text{op},k} \mathbf{D}, \\ \mathbf{e}_{y,jk}(\mathbf{x}_{\text{op}}) &= \mathbf{y}_{jk} - \mathbf{s}(\mathbf{T}_{\text{op},k} \mathbf{p}_{\text{op},j}), \end{aligned}$$

where we see that the only change from the SLAM case is that the $\mathbf{G}_{1,jk}$ matrix has a padding $\mathbf{0}$ to account for the fact that the ψ_k perturbation variable (associated with $\boldsymbol{\varpi}_k$) is not involved in the observation of landmark j from pose k . The part of the objective function associated with the measurements is then approximately

$$\begin{aligned} J_y(\mathbf{x}) &\approx \frac{1}{2} (\mathbf{e}_y(\mathbf{x}_{\text{op}}) - \mathbf{G}_1 \delta \mathbf{x}_1 - \mathbf{G}_2 \delta \mathbf{x}_2)^T \mathbf{R}^{-1} \\ &\quad \times (\mathbf{e}_y(\mathbf{x}_{\text{op}}) - \mathbf{G}_1 \delta \mathbf{x}_1 - \mathbf{G}_2 \delta \mathbf{x}_2), \quad (10.31) \end{aligned}$$

which is again quadratic in the perturbation variables, $\delta \mathbf{x}_1$ and $\delta \mathbf{x}_2$.

10.2.3 Batch Maximum a Posteriori Solution

With both the motion prior and the measurement terms in hand, we can write the full MAP objective function as

$$J(\mathbf{x}) = J_v(\mathbf{x}) + J_y(\mathbf{x}) \approx J(\mathbf{x}_{\text{op}}) - \mathbf{b}^T \delta \mathbf{x} + \delta \mathbf{x}^T \mathbf{A} \delta \mathbf{x}, \quad (10.32)$$

with

$$\mathbf{A} = \mathbf{H}^T \mathbf{W}^{-1} \mathbf{H}, \quad \mathbf{b} = \mathbf{H}^T \mathbf{W}^{-1} \mathbf{e}(\mathbf{x}_{\text{op}}), \quad (10.33)$$

and

$$\begin{aligned}\delta \mathbf{x} &= \begin{bmatrix} \delta \mathbf{x}_1 \\ \delta \mathbf{x}_2 \end{bmatrix}, \quad \mathbf{H} = \begin{bmatrix} \mathbf{F}^{-1} & \mathbf{0} \\ \mathbf{G}_1 & \mathbf{G}_2 \end{bmatrix}, \quad \mathbf{W} = \begin{bmatrix} \mathbf{Q} & \mathbf{0} \\ \mathbf{0} & \mathbf{R} \end{bmatrix}, \\ \mathbf{e}(\mathbf{x}_{\text{op}}) &= \begin{bmatrix} \mathbf{e}_v(\mathbf{x}_{\text{op}}) \\ \mathbf{e}_y(\mathbf{x}_{\text{op}}) \end{bmatrix}.\end{aligned}\quad (10.34)$$

The minimizing perturbation, $\delta \mathbf{x}^*$, is the solution to

$$\mathbf{A} \delta \mathbf{x}^* = \mathbf{b}. \quad (10.35)$$

As usual, we solve for $\delta \mathbf{x}^*$, then apply the optimal perturbations using the appropriate schemes,

$$\mathbf{T}_{\text{op},k} \leftarrow \exp \left(\epsilon_k^{*\wedge} \right) \mathbf{T}_{\text{op},k}, \quad (10.36a)$$

$$\boldsymbol{\varpi}_{\text{op},k} \leftarrow \boldsymbol{\varpi}_{\text{op},k} + \boldsymbol{\psi}_k^*, \quad (10.36b)$$

$$\mathbf{p}_{\text{op},j} \leftarrow \mathbf{p}_{\text{op},j} + \mathbf{D} \boldsymbol{\zeta}_j^*, \quad (10.36c)$$

and iterate to convergence. Similarly to the SLAM case, once converged we set $\hat{\mathbf{T}}_{v_k i} = \mathbf{T}_{\text{op},k}$, $\hat{\boldsymbol{\varpi}}_{v_k}^{v_k i} = \boldsymbol{\varpi}_{\text{op},k}$, and $\hat{\mathbf{p}}_i^{p_j i} = \mathbf{p}_{\text{op},j}$ at the last iteration as the final estimates for the vehicle poses, generalized velocity, and landmark positions of interest at the measurement times.

10.2.4 Exploiting Sparsity

Introducing the continuous-time motion priors does not destroy the nice sparsity of the discrete-time SLAM problem. We can see this by noting that

$$\begin{aligned}\mathbf{A} &= \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{12}^T & \mathbf{A}_{22} \end{bmatrix} = \mathbf{H}^T \mathbf{W}^{-1} \mathbf{H} \\ &= \begin{bmatrix} \mathbf{F}^{-T} \mathbf{Q}^{-1} \mathbf{F}^{-1} + \mathbf{G}_1^T \mathbf{R}^{-1} \mathbf{G}_1 & \mathbf{G}_1^T \mathbf{R}^{-1} \mathbf{G}_2 \\ \mathbf{G}_2^T \mathbf{R}^{-1} \mathbf{G}_1 & \mathbf{G}_2^T \mathbf{R}^{-1} \mathbf{G}_2 \end{bmatrix}.\end{aligned}\quad (10.37)$$

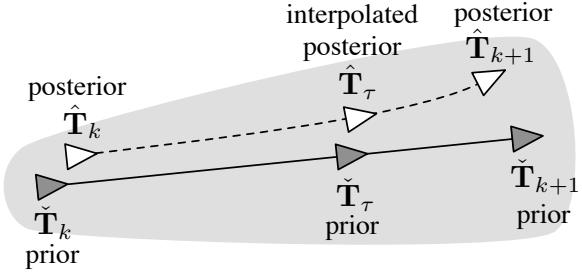
Compared to the SLAM problem, blocks \mathbf{A}_{12} and \mathbf{A}_{22} have not changed at all, showing that \mathbf{A} is still an arrowhead matrix with \mathbf{A}_{22} block-diagonal. We can thus exploit this sparsity to solve for the perturbations at each iteration efficiently using either the Schur or Cholesky methods.

The block \mathbf{A}_{11} looks very similar to the discrete-time SLAM case,

$$\mathbf{A}_{11} = \underbrace{\mathbf{F}^{-T} \mathbf{Q}^{-1} \mathbf{F}^{-1}}_{\text{prior}} + \underbrace{\mathbf{G}_1^T \mathbf{R}^{-1} \mathbf{G}_1}_{\text{measurements}}, \quad (10.38)$$

but recall that the \mathbf{G}_1 matrix is slightly different due the fact that we are estimating both pose and generalized velocity at each measurement time. Nevertheless, \mathbf{A}_{11} is still block-tridiagonal. Thus, we could choose to exploit the sparsity of \mathbf{A}_{11} rather than \mathbf{A}_{22} , if the number of poses were large compared to the number of landmarks, for example. In this

Figure 10.1 It is possible to use our continuous-time estimation framework to interpolate (for the mean and the covariance) between the main estimation times (i.e., measurement times).



case, the Cholesky method is preferred over the Schur one as we do not need to construct \mathbf{A}_{11}^{-1} , which is actually dense. Yan et al. (2014) explain how to use the sparse-GP method within the incremental approach of Kaess et al. (2008, 2012).

10.2.5 Interpolation

After we have solved for the state at the measurement times, we can also now use our Gaussian process framework to interpolate for the state at one or more query times. The situation is depicted in Figure 10.1, where our goal is to interpolate the posterior pose (and generalized velocity) at query time, τ .

Because we have deliberately estimated a Markovian state for our chosen SDE defining the prior, $\{\mathbf{T}_k, \boldsymbol{\varpi}_k\}$, we know that to interpolate at time τ , we need only consider the two measurements times on either side. Without loss of generality, we assume

$$t_k \leq \tau < t_{k+1}. \quad (10.39)$$

The difference between the posterior and the prior at times τ , t_k , and t_{k+1} we write as

$$\begin{aligned} \gamma_\tau &= \left[\ln \left(\hat{\mathbf{T}}_\tau \check{\mathbf{T}}_\tau^{-1} \right)^\vee \right], \quad \gamma_k = \left[\ln \left(\hat{\mathbf{T}}_k \check{\mathbf{T}}_k^{-1} \right)^\vee \right], \\ \gamma_{k+1} &= \left[\ln \left(\hat{\mathbf{T}}_{k+1} \check{\mathbf{T}}_{k+1}^{-1} \right)^\vee \right], \end{aligned} \quad (10.40)$$

where we note that the posterior values at the two measurement times come from the operating point values at the last iteration of the main MAP solution: $\{\hat{\mathbf{T}}_k, \hat{\boldsymbol{\varpi}}_k\} = \{\mathbf{T}_{\text{op},k}, \boldsymbol{\varpi}_{\text{op},k}\}$.

Using these definitions, we can go ahead and carry out state interpolation (for the mean) on $\mathfrak{se}(3) \times \mathbb{R}^3$ using the approach from Section 3.4:

$$\gamma_\tau = \Lambda(\tau) \gamma_k + \Psi(\tau) \gamma_{k+1}, \quad (10.41)$$

where

$$\Lambda(\tau) = \Phi(\tau, t_k) - \mathbf{Q}_\tau \Phi(t_{k+1}, \tau)^T \mathbf{Q}_{k+1}^{-1} \Phi(t_{k+1}, t_k), \quad (10.42a)$$

$$\Psi(\tau) = \mathbf{Q}_\tau \Phi(t_{k+1}, \tau)^T \mathbf{Q}_{k+1}^{-1}. \quad (10.42b)$$

We have all the required pieces to build these two matrices except \mathbf{Q}_τ , which is given by

$$\mathbf{Q}_\tau = \int_{t_k}^{\tau} \Phi(\tau, s) \mathbf{L} \mathbf{Q} \mathbf{L}^T \Phi(\tau, s)^T ds. \quad (10.43)$$

We can either integrate this numerically, or adopt the linear-motion simplification,

$$\check{\boldsymbol{\varpi}} = \begin{bmatrix} \check{\boldsymbol{\nu}} \\ \mathbf{0} \end{bmatrix}, \quad (10.44)$$

whereupon

$$\mathbf{Q}_\tau = \begin{bmatrix} \mathbf{Q}_{\tau,11} & \mathbf{Q}_{\tau,12} \\ \mathbf{Q}_{\tau,12}^T & \mathbf{Q}_{\tau,22} \end{bmatrix}, \quad (10.45a)$$

$$\begin{aligned} \mathbf{Q}_{\tau,11} &= \frac{1}{3} \Delta t_{\tau:k}^3 \mathbf{Q} + \frac{1}{8} \Delta t_{\tau:k}^4 \left(\check{\boldsymbol{\varpi}}^\wedge \mathbf{Q} + \mathbf{Q} \check{\boldsymbol{\varpi}}^{\wedge T} \right) \\ &\quad + \frac{1}{20} \Delta t_{\tau:k}^5 \check{\boldsymbol{\varpi}}^\wedge \mathbf{Q} \check{\boldsymbol{\varpi}}^{\wedge T}, \end{aligned} \quad (10.45b)$$

$$\mathbf{Q}_{\tau,12} = \frac{1}{2} \Delta t_{\tau:k}^2 \mathbf{Q} + \frac{1}{6} \Delta t_{\tau:k}^3 \check{\boldsymbol{\varpi}}^\wedge \mathbf{Q}, \quad (10.45c)$$

$$\mathbf{Q}_{\tau,22} = \Delta t_{\tau:k} \mathbf{Q}, \quad (10.45d)$$

$$\Delta t_{\tau:k} = \tau - t_k. \quad (10.45e)$$

Plugging this in, we can evaluate

$$\boldsymbol{\gamma}_\tau = \begin{bmatrix} \boldsymbol{\xi}_\tau \\ \boldsymbol{\eta}_\tau \end{bmatrix}, \quad (10.46)$$

and then compute the interpolated posterior on $SE(3) \times \mathbb{R}^3$ as

$$\hat{\mathbf{T}}_\tau = \exp(\boldsymbol{\xi}_\tau^\wedge) \check{\mathbf{T}}, \quad (10.47a)$$

$$\hat{\boldsymbol{\varpi}}_\tau = \check{\boldsymbol{\varpi}} + \boldsymbol{\eta}_\tau. \quad (10.47b)$$

The cost of this trajectory query is $O(1)$ since it only involves two measurement times in the interpolation equation. We can repeat this as many times as we like for different values of τ . A similar approach can be used to interpolate the covariance at the query time using the methods from Section 3.4.

10.2.6 Postscript

It is worth pointing out that while our underlying approach in this chapter considers a trajectory that is continuous in time, we are still

discretizing it in order to carry out the batch MAP solution at the measurement times and also the interpolation at the query times. The point is that we have a principled way to query the trajectory at any time of interest, not just the measurement times. Moreover, the interpolation scheme is chosen up front and provides the abilities to (i) smooth the solution based on a physically motivated prior and, (ii) carry out interpolation at any time of interest.

It is also worth noting that the Gaussian process approach taken in this chapter is quite different from the interpolation approach taken in Section 9.1.5. There we forced the motion between measurement times to have constant body-centric generalized velocity: it was a constraint-based interpolation method. Here we are defining a whole distribution of possible trajectories and encouraging the solution to find one that balances the prior with the measurements: this is a penalty-term approach. Both approaches have their merits.

Finally, it is worth making a point about the estimation philosophy used in this chapter. We have claimed that we presented a MAP method. However, in the nonlinear chapter, the MAP approach always linearized the motion model about the current MAP estimate. On the surface, it appears that we have done something slightly different in this chapter: to separate the desired nonlinear SDE into the nominal and perturbation components, we essentially linearized about the mean of the prior. We then built an error term for the motion prior and linearized that about the current MAP estimate. However, the other way to look at it is that we simply replaced the desired SDE with a slightly different one that was easier to work with and then applied the MAP approach for $SE(3)$. This is not the only way to apply the Gaussian process, continuous-time approach to estimation on $SE(3)$, but we hope it provides one useful example; Anderson and Barfoot (2015) provide an alternative.

References

- Absil, P A, Mahony, R, and Sepulchre, R. 2009. *Optimization on Matrix Manifolds*. Princeton University Press.
- Anderson, S, and Barfoot, T D. 2015 (28 September - 2 October). Full STEAM Ahead: Exactly Sparse Gaussian Process Regression for Batch Continuous-Time Trajectory Estimation on SE(3). In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.
- Anderson, S, Barfoot, T D, Tong, C H, and Särkkä, S. 2015. Batch Nonlinear Continuous-Time Trajectory Estimation as Exactly Sparse Gaussian Process Regression. *Autonomous Robots*, special issue on “Robotics Science and Systems”, **39**(3), 221–238.
- Arun, K S, Huang, T S, and Blostein, S D. 1987. Least-Squares Fitting of Two 3D Point Sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **9**(5), 698–700.
- Bailey, T, and Durrant-Whyte, H. 2006. Simultaneous Localisation and Mapping (SLAM): Part II State of the Art. *IEEE Robotics and Automation Magazine*, **13**(3), 108–117.
- Barfoot, T D, Forbes, J R, and Furgale, P T. 2011. Pose Estimation using Linearized Rotations and Quaternion Algebra. *Acta Astronautica*, **68**(1-2), 101–112.
- Barfoot, T D, Tong, C H, and Särkkä, S. 2014 (12-16 July). Batch Continuous-Time Trajectory Estimation as Exactly Sparse Gaussian Process Regression. In: *Proceedings of Robotics: Science and Systems (RSS)*.
- Barfoot, Timothy D, and Furgale, Paul T. 2014. Associating Uncertainty with Three-Dimensional Poses for use in Estimation Problems. *IEEE Transactions on Robotics*, **30**(3), 679–693.
- Bayes, Thomas. 1764. Essay towards solving a problem in the doctrine of chances. *Philosophical Transactions of the Royal Society of London*.
- Besl, P J, and McKay, N D. 1992. A Method for Registration of 3-D Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **14**(2), 239–256.
- Bierman, G J. 1974. Sequential Square Root Filtering and Smoothing of Discrete Linear Systems. *Automatica*, **10**(2), 147–158.
- Bishop, Christopher M. 2006. *Pattern Recognition and Machine Learning*. Secaucus, NJ, USA: Springer-Verlag New York, Inc.
- Box, M J. 1971. Bias in Nonlinear Estimation. *Journal of the Royal Statistical Society, Series B*, **33**(2), 171–201.
- Brookshire, J, and Teller, S. 2012 (July). Extrinsic Calibration from Per-Sensor Egomotion. In: *Proceedings of Robotics: Science and Systems*.
- Brown, D C. 1958. *A Solution to the General Problem of Multiple Station Analytical Stereotriangulation*. RCA-MTP Data Reduction Technical Report No. 43 (or AFMTC TR 58-8). Patrick Airforce Base, Florida.
- Bryson, A E. 1975. *Applied Optimal Control: Optimization, Estimation and Control*. Taylor and Francis.

- Chen, C S, Hung, Y P, and Cheng, J B. 1999. RANSAC-based DARCES: A new approach to fast automatic registration of partially overlapping range images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **21**(11), 1229–1234.
- Chirikjian, G S. 2009. *Stochastic Models, Information Theory, and Lie Groups: Classical Results and Geometric Methods*. Vol. 1-2. New York: Birkhauser.
- Chirikjian, G S, and Kyatkin, A B. 2001. *Engineering Applications of Noncommutative Harmonic Analysis: With Emphasis on Rotation and Motion Groups*. CRC Press.
- Chirikjian, G S, and Kyatkin, A B. 2016. *Harmonic Analysis for Engineers and Applied Scientists: Updated and Expanded Edition*. Dover Publications.
- Corke, P. 2011. *Robotics, Vision, and Control*. Springer Tracts in Advanced Robotics 73. Springer.
- Davenport, P B. 1965. *A Vector Approach to the Algebra of Rotations with Applications*. Tech. rept. X-546-65-437. NASA.
- de Ruiter, A H J, and Forbes, J R. 2013. On the Solution of Wahba’s Problem on SO(n). *Journal of the Astronautical Sciences*, **60**(1), 1–31.
- D’Eleuterio, G M T. 1985 (June). *Multibody Dynamics for Space Station Manipulators: Recursive Dynamics of Topological Chains*. Tech. rept. SS-3. Dynacon Enterprises Ltd.
- Devlin, Keith. 2008. *The Unfinished Game: Pascal, Fermat, and the Seventeenth-Century Letter that Made the World Modern*. Basic Book.
- Dudek, G, and Jenkin, M. 2010. *Computational Principles of Mobile Robotics*. Cambridge University Press.
- Durrant-Whyte, H, and Bailey, T. 2006. Simultaneous Localisation and Mapping (SLAM): Part I The Essential Algorithms. *IEEE Robotics and Automation Magazine*, **11**(3), 99–110.
- Dyce, M. 2013. Canada between the photograph and the map: Aerial photography, geographical vision and the state. *Journal of Historical Geography*, **39**, 69–84.
- Fischler, M, and Bolles, R. 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of ACM*, **24**(6), 381–395.
- Furgale, P T. 2011. *Extensions to the Visual Odometry Pipeline for the Exploration of Planetary Surfaces*. Ph.D. thesis, University of Toronto.
- Furgale, P T, Tong, C H, Barfoot, T D, and Sibley, G. 2015. Continuous-Time Batch Trajectory Estimation Using Temporal Basis Functions. *International Journal of Robotics Research*, **34**(14), 1688–1710.
- Green, B F. 1952. The Orthogonal Approximation of an Oblique Structure in Factor Analysis. *Psychometrika*, **17**(4), 429–440.
- Hartley, R, and Zisserman, A. 2000. *Multiple View Geometry in Computer Vision*. Cambridge University Press.
- Hertzberg, C, Wagner, R, Frese, U, and Schröder, L. 2013. Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds. *Information Fusion*, **14**(1), 57 – 77.
- Holland, P W, and Welsch, R E. 1977. Robust Regression Using Iteratively Reweighted Least-Squares. *Communications in Statistics – Theory and Methods*, **6**(9), 813–827.
- Horn, B K P. 1987a. Closed-Form Solution of Absolute Orientation using Orthonormal Matrices. *Journal of the Optical Society of America A*, **5**(7), 1127–1135.
- Horn, B K P. 1987b. Closed-Form Solution of Absolute Orientation using Unit Quaternions. *Journal of the Optical Society of America A*, **4**(4), 629–642.
- Hughes, Peter C. 1986. *Spacecraft Attitude Dynamics*. Dover.

- Jazwinski, A H. 1970. *Stochastic Processes and Filtering Theory*. Academic, New York.
- Julier, S, and Uhlmann, J. 1996. *A General Method for Approximating Nonlinear Transformations of Probability Distributions*. Tech. rept. Robotics Research Group, University of Oxford.
- Kaess, M, Ranganathan, A, and Dellaert, R. 2008. iSAM: Incremental Smoothing and Mapping. *IEEE TRO*, **24**(6), 1365–1378.
- Kaess, M, Johannsson, H, Roberts, R, Ila, V, Leonard, J J, and Dellaert, F. 2012. iSAM2: Incremental Smoothing and Mapping Using the Bayes Tree. *IJRR*, **31**(2), 217–236.
- Kalman, R E. 1960a. Contributions to the Theory of Optimal Control. *Boletin de la Sociedad Matematica Mexicana*, **5**, 102–119.
- Kalman, R E. 1960b. A New Approach to Linear Filtering and Prediction Problems. *Trans. ASME, Journal of Basic Engineering*, **82**, 35–45.
- Kelly, Alonzo. 2013. *Mobile Robotics: Mathematics, Models, and Methods*. Cambridge University Press.
- Klarsfeld, S, and Oteo, J A. 1989. The Baker-Campbell-Hausdorff formula and the convergence of the Magnus expansion. *Journal of Phys. A: Math. Gen.*, **22**, 4565–4572.
- Lee, T, Leok, M, and McClamroch, N H. 2008. Global Symplectic Uncertainty Propagation on SO(3). Pages 61–66 of: *Proceedings of the 47th IEEE Conference on Decision and Control*.
- Long, A W, Wolfe, K C, Mashner, M J, and Chirikjian, G S. 2012. The Banana Distribution is Gaussian: A Localization Study with Exponential Coordinates. In: *Proceedings of Robotics: Science and Systems*.
- Lowe, D G. 2004. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, **60**(2), 91–110.
- Lu, F, and Milios, E. 1997. Globally Consistent Range Scan Alignment for Environment Mapping. *Auton. Robots*, **4**(4), 333–349.
- MacTavish, K A, and Barfoot, T D. 2015 (3-5 June). At All Costs: A Comparison of Robust Cost Functions for Camera Correspondence Outliers. Pages 62–69 of: *Proceedings of the 12th Conference on Computer and Robot Vision (CRV)*.
- Madow, William F. 1949. On the Theory of Systematic Sampling, II. *Annals of Mathematical Statistics*, **30**, 333–354.
- Mahalanobis, P. 1936. On the Generalized Distance in Statistics. Pages 49–55 of: *Proceedings of the National Institute of Science*, vol. 2.
- Matthies, L, and Shafer, S A. 1987. Error Modeling in Stereo Navigation. *IEEE Journal of Robotics and Automation*, **3**(3), 239–248.
- Maybeck, Peter S. 1994. *Stochastic Models, Estimation and Control*. Navtech Book and Software Store.
- McGee, L A, and Schmidt, S F. 1985 (November). *Discovery of the Kalman Filter as a Practical Tool for Aerospace and Industry*. Tech. rept. NASA-TM-86847. NASA.
- Murray, R M, Li, Z, and Sastry, S. 1994. *A Mathematical Introduction to Robotic Manipulation*. CRC Press.
- Papoulis, Athanasios. 1965. *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill Book Company, New York.
- Peretroukhin, V, Vega-Brown, W, Roy, N, and Kelly, J. 2016 (16-21 May). PROBE-GK: Predictive Robust Estimation Using Generalized Kernels. Pages 817–824 of: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.
- Rasmussen, C E, and Williams, C K I. 2006. *Gaussian Processes for Machine Learning*. Cambridge, MA: MIT Press.

- Rauch, H E, Tung, F, and Striebel, C T. 1965. Maximum Likelihood Estimates of Linear Dynamic Systems. *AIAA Journal*, **3**(8), 1445–1450.
- Särkkä, S. 2006. *Recursive Bayesian Inference on Stochastic Differential Equations*. Ph.D. thesis, Helsinki University of Technology.
- Särkkä, S. 2013. *Bayesian Filtering and Smoothing*. Cambridge University Press.
- Sastry, S. 1999. *Nonlinear Systems: Analysis, Stability, and Control*. New York: Springer.
- Shannon, Claude E. 1948. A Mathematical Theory of Communication. *The Bell System Technical Journal*, **27**, 379–423, 623–656.
- Sherman, J., and Morrison, W J. 1949. Adjustment of an Inverse Matrix Corresponding to Changes in the Elements of a Given Column or Given Row of the Original Matrix. *Annals of Mathematics and Statistics*, **20**, 621.
- Sherman, J., and Morrison, W J. 1950. Adjustment of an Inverse Matrix Corresponding to a Change in One Element of a Given Matrix. *Annals of Mathematics and Statistics*, **21**, 124–127.
- Sibley, G. 2006. *A Sliding Window Filter for SLAM*. Tech. rept. University of Southern California.
- Sibley, G., Sukhatme, G., and Matthies, L. 2006. The Iterated Sigma Point Kalman Filter with Applications to Long-Range Stereo. In: *Proceedings of Robotics: Science and Systems*.
- Sibley, Gabe. 2007. *Long Range Stereo Data-Fusion From Moving Platforms*. Ph.D. thesis, University of Southern California.
- Simon, D. 2006. *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*. Wiley-Interscience.
- Smith, P, Drummond, T, and Roussopoulos, K. 2003. Computing MAP Trajectories by Representing, Propagating, and Combining PDFs Over Groups. In: *Proceedings of the IEEE International Conference on Computer Vision*.
- Smith, Randall C., Self, Matthew, and Cheeseman, Peter. 1990. Estimating Uncertain Spatial Relationships in Robotics. Pages 167–193 of: Cox, Ingemar J., and Wilfong, Gordon T. (eds), *Autonomous Robot Vehicles*. New York: Springer Verlag.
- Stengel, R F. 1994. *Optimal Control and Estimation*. Dover Publications Inc.
- Stillwell, J. 2008. *Naive Lie Theory*. Springer.
- Stuepnagel, J. 1964. On the Parameterization of the Three-Dimensional Rotation Group. *SIAM Review*, **6**(4), 422–430.
- Su, S F, and Lee, C S G. 1991. Uncertainty manipulation and propagation and verification of applicability of actions in assembly tasks. Pages 2471–2476 of: *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 3.
- Su, S F, and Lee, C S G. 1992. Manipulation and propagation of uncertainty and verification of applicability of actions in assembly tasks. *IEEE Transactions on Systems, Man and Cybernetics*, **22**(6), 1376–1389.
- Thrun, S., and Montemerlo, M. 2005. The GraphSLAM Algorithm With Applications to Large-Scale Mapping of Urban Structures. *International Journal on Robotics Research*, **25**(5/6), 403–430.
- Thrun, Sebastian, Fox, Dieter, Burgard, Wolfram, and Dellaert, Frank. 2001. Robust Monte Carlo localization for mobile robots. *Artificial Intelligence*, **128**(1–2), 99–141.
- Thrun, Sebastian, Burgard, Wolfram, and Fox, Dieter. 2006. *Probabilistic Robotics*. MIT Press.
- Tong, C H, Furgale, P T, and Barfoot, T D. 2013. Gaussian Process Gauss-Newton for Non-Parametric Simultaneous Localization and Mapping. *International Journal of Robotics Research*, **32**(5), 507–525.

- Triggs, W, McLauchlan, P, Hartley, R, and Fitzgibbon, A. 2000. Bundle Adjustment: A Modern Synthesis. Pages 298–375 of: Triggs, W, Zisserman, A, and Szeliski, R (eds), *Vision Algorithms: Theory and Practice*. LNCS. Springer Verlag.
- Umeyama, S. 1991. Least-Squares Estimation of Transformation Parameters Between Two Point Patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **13**(4), 376–380.
- Wahba, G. 1965. A Least-Squares Estimate of Spacecraft Attitude. *SIAM Review*, **7**(3), 409.
- Wang, Y, and Chirikjian, G S. 2006. Error Propagation on the Euclidean Group With Applications to Manipulator Kinematics. *IEEE Transactions on Robotics*, **22**(4), 591–602.
- Wang, Y, and Chirikjian, G S. 2008. Nonparametric Second-order Theory of Error Propagation on Motion Groups. *International Journal of Robotics Research*, **27**(11), 1258–1273.
- Wolfe, K, Mashner, M, and Chirikjian, G. 2011. Bayesian Fusion on Lie Groups. *Journal of Algebraic Statistics*, **2**(1), 75–97.
- Woodbury, M A. 1950. *Inverting Modified Matrices*. Tech. rept. 42. Statistical Research Group, Princeton University.
- Yan, X, Indelman, V, and Boots, B. 2014. Incremental Sparse GP Regression for Continuous-time Trajectory Estimation and Mapping. In: *Proceedings of the NIPS Workshop on Autonomously Learning Robots*.
- Zhang, Zhengyou. 1997. Parameter Estimation Techniques: A Tutorial with Application to Conic Fitting. *Image and Vision Computing*, **15**(1), 59–76.

Index

- adaptive estimation, 166
adjoint, 226, 227
affine transformation, 203
algebra, 217
Apianus, Petrus, xv
arrowhead matrix, 345, 346
axiom of total probability, 9
BA, *see* bundle adjustment
Baker, Henry Frederick, 231
Baker-Campbell-Hausdorff, 231, 232, 234, 237, 247, 248, 274, 281, 326, 331, 360, 393
Bayes filter, xv, 3, 68, 91, 97–103, 107, 115, 116, 127, 142
Bayes’ rule, 3, 10, 33, 40, 50, 94, 98
Bayes, Thomas, 11
Bayesian, 9
Bayesian inference, 11, 24, 39, 44, 46, 68–70, 91, 92, 135, 137, 143, 146
BCH, *see* Baker-Campbell-Hausdorff
belief function, 97
Bernoulli numbers, 232
Bernoulli, Jakob, 232, 243
Bessel’s correction, 12
Bessel, Friedrich Wilhelm, 12
best linear unbiased estimate, 70
biased, 103, 139
BLUE, *see* best linear unbiased estimate
bundle adjustment, 337, 348, 351, 352, 354, 355
camera, 199
Campbell, John Edward, 231
Cauchy cost function, 164
Cauchy product, 243
Cauchy, Baron Augustin-Louis, 243
causal, 58
Cayley-Hamilton theorem, 49
Cayley-Rodrigues parameters, 182
Cholesky decomposition, 52–55, 87, 90, 110, 111, 119, 120, 124, 129, 276, 289, 334, 335, 345, 347, 354, 355, 365
Cholesky smoother, 53
Cholesky, André-Louis, 52
consistent, 71, 152
continuous time, xvi, 4, 9, 32, 33, 37, 74, 88, 91, 96, 143, 147, 320, 321, 357, 358, 362, 363, 365–368
covariance estimation, 166
covariance matrix, 12
Cramér, Harold, 14
Cramér-Rao lower bound, 14, 15, 31, 32, 70, 72, 118
CRLB, *see* Cramér-Rao lower bound
cross product, 175, 176
cubic Hermite polynomial, 86
curvature, 196
DARCES, *see* data-aligned rigidity-constrained exhaustive search
data association, 151, 159
data-aligned rigidity-constrained exhaustive search, 161
Dirac, Paul Adrien Maurice, 33
directional derivative, 247, 393
discrete time, 28, 32, 37, 51, 58, 74, 80, 87, 88, 96, 97, 127, 143, 147, 149, 277, 319–322, 325, 357, 363, 365
disparity, 208
dot product, 175, 177
early estimation milestones, 3
EKF, *see* extended Kalman filter
epipolar constraint, 203
epipolar line, 203
essential matrix (of computer vision), 201
estimate, 38
estimation, *see* state estimation
Euler parameters, *see* unit-length quaternions
Euler’s rotation theorem, 180, 216
Euler, Leonhard, 178
exponential map, 219
extended Kalman filter, 70, 91, 100, 101, 103, 104, 106, 107, 109, 115, 118, 121, 122, 124–127, 134, 142, 143, 149, 297, 319, 321–324
exteroceptive, 3
extrinsic sensor parameters, 199

- factor graph, 355
 Faulhaber's formula, 243
 Faulhaber, Johann, 243
 filter, 59
 Fisher, Sir Ronald Aylmer, 15
 fixed-internal smoother, 43, 51
 focal length, 200
 Frenet, Jean Frédéric, 196
 Frenet-Serret frame, 196, 198, 212
 frequentist, 9
 Frobenius norm, 279, 291
 frontal projection model, 199
 fundamental matrix (of computer vision), 203
 fundamental matrix (of control theory), 145, 264
 Gauss, Carl Friedrich, 2, 3
 Gauss-Newton optimization, 129–134, 138, 139, 142, 249, 250, 254, 282, 283, 318, 319, 326–329, 333, 342, 343, 345
 Gaussian estimator, 50, 63, 107
 Gaussian filter, 115
 Gaussian inference, 19
 Gaussian noise, 1, 2, 70, 88, 92, 100, 101, 151, 152, 155, 321, 339
 Gaussian probability density function, 9, 13–16, 18–20, 22, 24, 26, 28–31, 33, 60, 63, 93, 99–102, 104, 105, 107–113, 115, 119, 120, 124, 126, 136, 146, 268, 288, 330
 Gaussian process, xvi, 4, 9, 32, 33, 74–77, 81, 85, 87, 143, 145–148, 357, 358, 363, 366
 Gaussian random variable, 9, 16, 20, 24, 37, 266, 267, 270
 Geman-McClure cost function, 164
 generalized mass matrix, 318
 generalized velocity, 258
 Gibbs vector, 182
 Gibbs, Josiah Willard, 182
 global positioning system, 4, 159–161
 GP, *see* Gaussian process
 GPS, *see* global positioning system
 group, 216
 Hamilton, Sir William Rowan, 181
 Hausdorff, Felix, 231
 Heaviside step function, 79
 Heaviside, Oliver, 182
 Hermite basis function, 87
 Hermite, Charles, 86
 homogeneous coordinates, 193, 246, 285
 homography matrix, 205
 ICP, *see* iterative closest point
 identity matrix, 175
 IEKF, *see* iterated extended Kalman filter
 improper rotation, 216
 IMU, *see* inertial measurement unit
 inconsistent, 103
 inertial measurement unit, 209, 211, 213
 information form, 56, 57, 67
 information matrix, 53
 information vector, 50
 injection, 268
 injective, 20
 inner product, *see* dot product
 interoceptive, 3
 interpolation matrix, 351
 intrinsic parameter matrix, 202
 inverse covariance form, *see* information form
 inverse-Wishart distribution, 166
 IRLS, *see* iterated reweighted least squares
 ISPKF, *see* iterated sigmapoint Kalman filter
 Isserlis' theorem, 16, 288
 Isserlis, Leon, 16
 Itō calculus, 77
 Itō, Kiyoshi, 77
 iterated extended Kalman filter, 105–107, 109, 124–127, 136, 137, 142, 146, 148
 iterated sigmapoint Kalman filter, 123, 125–127
 iterative closest point, 297, 298
 iteratively reweighted least squares, 165
 Jacobi's formula, 222
 Jacobi, Gustav Jacob, 218
 Jacobian, 224, 233, 234, 248
 John Harrison, 2
 joint probability density function, 10
 Kálmán, Rudolf Emil, 2
 Kōwa, Seki, 232
 Kalman filter, xv, 2, 3, 37, 58, 63, 68–70, 72, 153, 154, 159
 Kalman gain, 67
 kernel matrix, 75, 80
 KF, *see* Kalman filter
 kinematics, 184, 197, 198, 255, 256, 258, 259, 261–263, 265, 266, 274, 320, 321, 323
 kurtosis, 12, 115
 law of large numbers, 108
 Levenberg-Marquardt, 132, 254
 LG, *see* linear-Gaussian
 Lie algebra, 217
 Lie derivative, 248
 Lie group, *see* matrix Lie group
 Lie product formula, 232
 Lie, Marius Sophus, 215
 lifted form, 39, 44, 78
 line search, 132, 254

- linear time-invariant, 83, 359
 linear time-varying, 77, 81, 144, 264, 266
 linear, time-varying, 37
 linear-Gaussian, 38, 39, 43, 44, 46, 59, 61, 64, 73, 98, 159
 Lovelace, Ada, 232
 LTI, *see* linear time-invariant
 LTV, *see* linear time-varying
 M-estimation, 163
 Möbius, Augustus Ferdinand, 193
 Mahalanobis, 282
 Mahalanobis distance, 28, 41
 Mahalanobis, Prasanta Chandra, 28
 many-to-one, 220
 MAP, *see* maximum a posteriori
 marginalization, 11
 Markov property, 64, 97
 matrix inversion lemma, *see*
 Sherman-Morrison-Woodbury
 matrix Lie group, 215, 216
 maximum a posteriori, 39, 40, 63, 64, 69, 88, 89, 91, 94, 95, 106, 107, 125–128, 137, 138, 148, 151, 321, 322, 325, 326, 352, 364, 368
 maximum likelihood, 137, 138, 149, 151, 330, 331, 339, 342, 351
 mean, 11, 15
 mean rotation, 269
 ML, *see* maximum likelihood
 Monte Carlo, 100, 108, 279, 290
 Moore-Penrose pseudoinverse, *see*
 pseudoinverse
 mutual information, 14, 30
 NASA, *see* National Aeronautics and Space Administration
 National Aeronautics and Space Administration, 3, 100
 Newton's method, 129
 NLNG, *see* nonlinear, non-Gaussian
 non-commutative group, 182, 188, 215
 nonlinear, non-Gaussian, 91, 96, 97
 normalized image coordinates, 200
 normalized product, 13, 22
 observability, 2, 49, 156
 observability matrix, 49
 onto, 220
 optical axis, 199
 outlier, 151, 161, 162
 particle filter, 91, 115–118
 PDF, *see* probability density function
 point-cloud alignment, 297
 point-clouds, 297
 Poisson's equation, 186
 Poisson, Siméon Denis, 186
 pose-graph relaxation, 329
 poses, 173, 192, 216, 218, 222, 234, 239, 245, 251, 258, 265, 270, 273, 280
 posterior, 11, 38
 power spectral density martrix, 77
 power spectral density matrix, 33
 prior, 11, 38
 probability, 9
 probability density function, 9–12, 14–16, 19, 22–26, 28–30, 34, 95, 97–99, 101, 104, 107–113, 115, 116, 148, 268, 269, 276, 282
 probability distributions, 10
 proper rotation, 216
 pseudoinverse, 43
 quaternion, 298
 RAE, *see* range-azimuth-elevation
 random sample consensus, 162, 168, 169, 297, 305
 random variable, 9
 range-azimuth-elevation, 208, 209
 RANSAC, *see* random sample consensus
 Rao, Calyampudi Radhakrishna, 14
 Rauch, Herbert E., 55
 Rauch-Tung-Striebel smoother, 3, 51, 55, 58
 realization, 12, 14, 38
 reference frame, 174
 robust cost, 164
 rotary reflection, 216
 rotation matrix, 176, *see also* rotations
 rotations, 173, 215, 220, 232, 237, 240, 242, 247, 255, 261, 267
 RTS, *see* Rauch-Tung-Striebel
 sample covariance, 12
 sample mean, 12
 Schmidt, Stanley F., 100
 Schur complement, 19, 65, 345, 346, 348, 354, 355, 365, 366
 Schur, Issai, 19
 SDE, *see* stochastic differential equation
 Serret, Joseph Alfred, 196
 Shannon information, 14, 28, 29, 33
 Shannon, Claude Elwood, 14
 Sherman-Morrison-Woodbury, 23, 45, 55–57, 75, 123, 136, 137, 147
 sigmapoint, 110, 115, 120
 sigmapoint Kalman filter, 91, 118, 121, 122, 124–127
 sigmapoint transformation, 110, 113, 114, 119, 120, 148, 272, 276, 279, 285, 288
 simultaneous localization and mapping, 342, 352, 355, 363–365
 simultaneous trajectory estimation and mapping, 362
 singular-value decomposition, 305

- skewness, 12, 115
- SLAM, *see* simultaneous localization and mapping
- sliding-window filter, 142
- smoother, 59
- SMW, *see*
 - Sherman-Morrison-Woodbury
- SP, *see* sigmapoint
- sparse bundle adjustment, 345
- special Euclidean group, *see also* poses, 216
- special orthogonal group, *see also* rotations, 215
- SPKF, *see* sigmapoint Kalman filter
- state, 1, 38
- state estimation, 1, 4, 38
- state transition matrix, 264
- statistical moments, 11
- statistically independent, 10, 12, 20, 30
- STEAM, *see* simultaneous trajectory estimation and mapping
- stereo baseline, 206
- stereo camera, 92
- stochastic differential equation, 144, 266, 358, 360, 366, 368
- Striebel, Charlotte T., 55
- surjective-only, 220
- SWF, *see* sliding-window filter
- Sylvester's determinant theorem, 30
- Sylvester, James Joseph, 30
- tangent space, 218
- taxonomy of filtering methods, 127
- torsion, 196
- transformation matrix, 193, *see also* poses
- transition function, 77
- transition matrix, 38, 78
- Tung, Frank F., 55
- UKF, *see* sigmapoint Kalman filter
- unbiased, 14, 71, 152
- uncertainty ellipsoid, 29
- uncorrelated, 12, 20
- unimodular, 238
- unit-length quaternions, 180
- unscented Kalman filter, *see* sigmapoint Kalman filter
- variance, 15
- vector, 174
- vectrix, 174
- white noise, 33

Appendix A

Supplementary Material

Material generated since the first edition.

A.1 Lie Group Tools

A.1.1 $SE(3)$ Derivative

On occasion, we may want to take the derivative of the product of a 6×6 transformation matrix and a 6×1 column, with respect to the 6×1 pose variable.

To do this, we can start by taking the derivative with respect to a single element of $\xi = (\xi_1, \xi_2, \xi_3, \xi_4, \xi_5, \xi_6)$. Applying the definition of a derivative along the $\mathbf{1}_i$ direction, we have

$$\frac{\partial(\mathcal{T}(\xi)\mathbf{x})}{\partial\xi_i} = \lim_{h\rightarrow 0} \frac{\exp((\xi + h\mathbf{1}_i)^\wedge)\mathbf{x} - \exp(\xi^\wedge)\mathbf{x}}{h}, \quad (\text{A.1})$$

which we previously referred to as a *directional derivative*. Since we are interested in the limit of h infinitely small, we can use the approximate BCH formula to write

$$\begin{aligned} \exp((\xi + h\mathbf{1}_i)^\wedge) &\approx \exp((\mathcal{J}(\xi)h\mathbf{1}_i)^\wedge)\exp(\xi^\wedge) \\ &\approx (\mathbf{1} + h(\mathcal{J}(\xi)\mathbf{1}_i)^\wedge)\exp(\xi^\wedge), \end{aligned} \quad (\text{A.2})$$

where $\mathcal{J}(\xi)$ is the (left) Jacobian of $SE(3)$, evaluated at ξ . Plugging this back into (A.1), we find that

$$\frac{\partial(\mathcal{T}(\xi)\mathbf{x})}{\partial\xi_i} = (\mathcal{J}(\xi)\mathbf{1}_i)^\wedge \mathcal{T}(\xi)\mathbf{x} = -(\mathcal{T}(\xi)\mathbf{x})^\wedge \mathcal{J}(\xi)\mathbf{1}_i. \quad (\text{A.3})$$

Stacking the six directional derivatives alongside one another provides the desired Jacobian:

$$\frac{\partial(\mathcal{T}(\xi)\mathbf{x})}{\partial\xi} = -(\mathcal{T}(\xi)\mathbf{x})^\wedge \mathcal{J}(\xi). \quad (\text{A.4})$$

A.2 Kinematics

A.2.1 $SO(3)$ Jacobian Identity

An important identity that is used frequently in rotational kinematics is

$$\dot{\mathbf{J}}(\phi) - \boldsymbol{\omega}^\wedge \mathbf{J}(\phi) \equiv \frac{\partial \boldsymbol{\omega}}{\partial \phi}, \quad (\text{A.5})$$

where the relationship between angular velocity and the rotational parameter derivative is

$$\boldsymbol{\omega} = \mathbf{J}(\phi)\dot{\phi}. \quad (\text{A.6})$$

Beginning with the right-hand side, we have

$$\begin{aligned} \frac{\partial \boldsymbol{\omega}}{\partial \phi} &= \frac{\partial}{\partial \phi} (\mathbf{J}(\phi)\dot{\phi}) = \frac{\partial}{\partial \phi} \left(\underbrace{\int_0^1 \mathbf{C}(\phi)^\alpha d\alpha}_{\mathbf{J}(\phi)} \dot{\phi} \right) \\ &= \int_0^1 \frac{\partial}{\partial \phi} (\mathbf{C}(\alpha\phi)\dot{\phi}) d\alpha = - \int_0^1 (\mathbf{C}(\alpha\phi)\dot{\phi})^\wedge \alpha \mathbf{J}(\alpha\phi) d\alpha. \end{aligned} \quad (\text{A.7})$$

Noting that

$$\frac{d}{d\alpha} (\alpha \mathbf{J}(\alpha\phi)) = \mathbf{C}(\alpha\phi), \quad \int \mathbf{C}(\alpha\phi) d\alpha = \alpha \mathbf{J}(\alpha\phi), \quad (\text{A.8})$$

we can then integrate by parts to see that

$$\begin{aligned} \frac{\partial \boldsymbol{\omega}}{\partial \phi} &= - \underbrace{(\alpha \mathbf{J}(\alpha\phi)\dot{\phi})^\wedge \alpha \mathbf{J}(\alpha\phi)}_{\boldsymbol{\omega}^\wedge \mathbf{J}(\phi)} \Big|_{\alpha=0}^{\alpha=1} + \int_0^1 \underbrace{(\alpha \mathbf{J}(\alpha\phi)\dot{\phi})^\wedge \mathbf{C}(\alpha\phi)}_{\dot{\mathbf{C}}(\alpha\phi)} d\alpha \\ &= -\boldsymbol{\omega}^\wedge \mathbf{J}(\phi) + \frac{d}{dt} \underbrace{\int_0^1 \mathbf{C}(\phi)^\alpha d\alpha}_{\mathbf{J}(\phi)} = \dot{\mathbf{J}}(\phi) - \boldsymbol{\omega}^\wedge \mathbf{J}(\phi), \end{aligned} \quad (\text{A.9})$$

which is the desired result.

A.2.2 $SE(3)$ Jacobian Identity

We can derive a similar identity for pose kinematics:

$$\dot{\mathcal{J}}(\xi) - \boldsymbol{\varpi}^\wedge \mathcal{J}(\xi) \equiv \frac{\partial \boldsymbol{\varpi}}{\partial \xi}, \quad (\text{A.10})$$

where the relationship between generalized velocity and the pose parameter derivative is

$$\boldsymbol{\varpi} = \mathcal{J}(\xi)\dot{\xi}. \quad (\text{A.11})$$

Beginning with the right-hand side, we have

$$\begin{aligned}\frac{\partial \varpi}{\partial \xi} &= \frac{\partial}{\partial \xi} (\mathcal{J}(\xi) \dot{\xi}) = \frac{\partial}{\partial \xi} \left(\underbrace{\int_0^1 \mathcal{T}(\xi)^\alpha d\alpha}_{\mathcal{J}(\xi)} \dot{\xi} \right) \\ &= \int_0^1 \frac{\partial}{\partial \xi} (\mathcal{T}(\alpha \xi) \dot{\xi}) d\alpha = - \int_0^1 (\mathcal{T}(\alpha \xi) \dot{\xi})^\wedge \alpha \mathcal{J}(\alpha \xi) d\alpha.\end{aligned}\quad (\text{A.12})$$

Noting that

$$\frac{d}{d\alpha} (\alpha \mathcal{J}(\alpha \xi)) = \mathcal{T}(\alpha \xi), \quad \int \mathcal{T}(\alpha \xi) d\alpha = \alpha \mathcal{J}(\alpha \xi), \quad (\text{A.13})$$

we can then integrate by parts to see that

$$\begin{aligned}\frac{\partial \varpi}{\partial \xi} &= - \underbrace{\left(\alpha \mathcal{J}(\alpha \xi) \dot{\xi} \right)^\wedge \alpha \mathcal{J}(\alpha \xi) \Big|_{\alpha=0}^{1=1}}_{\varpi^\wedge \mathcal{J}(\xi)} + \int_0^1 \underbrace{(\alpha \mathcal{J}(\alpha \xi) \dot{\xi})^\wedge \mathcal{T}(\alpha \xi)}_{\dot{\mathcal{T}}(\alpha \xi)} d\alpha \\ &= -\varpi^\wedge \mathcal{J}(\xi) + \frac{d}{dt} \underbrace{\int_0^1 \mathcal{T}(\xi)^\alpha d\alpha}_{\mathcal{J}(\xi)} = \dot{\mathcal{J}}(\xi) - \varpi^\wedge \mathcal{J}(\xi),\end{aligned}\quad (\text{A.14})$$

which is the desired result.

A.3 Smoothers

A.3.1 Posterior Covariance in the Cholesky Smoother

In the development of the Cholesky smoother in 3.2.2, we did not explain how to extract the posterior covariance of the estimate without inverting the full matrix. We had defined the block-tridiagonal inverse of the posterior covariance as

$$\hat{\mathbf{P}}^{-1} = \mathbf{L} \mathbf{L}^T, \quad (\text{A.15})$$

where the non-zero sub-blocks of \mathbf{L} were

$$\mathbf{L} = \begin{bmatrix} \mathbf{L}_0 & & & & & \\ \mathbf{L}_{10} & \mathbf{L}_1 & & & & \\ & \mathbf{L}_{21} & \mathbf{L}_2 & & & \\ & & \ddots & \ddots & & \\ & & & \mathbf{L}_{K-1,K-2} & \mathbf{L}_{K-1} & \\ & & & & \mathbf{L}_{K,K-1} & \mathbf{L}_K \end{bmatrix}. \quad (\text{A.16})$$

The covariance is therefore

$$\hat{\mathbf{P}} = \mathbf{L}^{-T} \mathbf{L}^{-1}, \quad (\text{A.17})$$

where

$$\mathbf{L}^{-1} = \begin{bmatrix} \mathbf{L}_0^{-1} & & & & & \\ -\mathbf{L}_1^{-1}\mathbf{L}_{10}\mathbf{L}_0^{-1} & \ddots & & & & \\ \ddots & \ddots & \ddots & & & \\ \ddots & \ddots & & \mathbf{L}_{K-2}^{-1} & & \\ \ddots & \ddots & & -\mathbf{L}_{K-1}^{-1}\mathbf{L}_{K-1,K-2}\mathbf{L}_{K-2}^{-1} & \mathbf{L}_{K-1}^{-1} & \\ \ddots & \ddots & & \mathbf{L}_K^{-1}\mathbf{L}_{K,K-1}\mathbf{L}_{K-1}^{-1}\mathbf{L}_{K-1,K-2}\mathbf{L}_{K-2}^{-1} & -\mathbf{L}_K^{-1}\mathbf{L}_{K,K-1}\mathbf{L}_{K-1}^{-1} & \mathbf{L}_K^{-1} \end{bmatrix}. \quad (\text{A.18})$$

Unfortunately, the lower triangle of \mathbf{L}^{-1} is now dense and so is $\hat{\mathbf{P}}$. Luckily, we can still solve for only the main block diagonal (and additional diagonals upto some block bandwidth) in $O(K)$ time using a backward recursion. We will show this for the main diagonal and one additional diagonal, which is necessary for the covariance interpolation formula in (3.198b).

The blocks of $\hat{\mathbf{P}}$ are

$$\hat{\mathbf{P}} = \begin{bmatrix} \hat{\mathbf{P}}_0 & \hat{\mathbf{P}}_{10}^T & \ddots & \ddots & \ddots & \ddots \\ \hat{\mathbf{P}}_{10} & \hat{\mathbf{P}}_1 & \hat{\mathbf{P}}_{21}^T & \ddots & \ddots & \ddots \\ \ddots & \hat{\mathbf{P}}_{21} & \ddots & \ddots & \ddots & \ddots \\ \ddots & \ddots & \ddots & \hat{\mathbf{P}}_{K-2} & \hat{\mathbf{P}}_{K-1,K-2}^T & \ddots \\ \ddots & \ddots & \ddots & \hat{\mathbf{P}}_{K-1,K-2} & \hat{\mathbf{P}}_{K-1} & \hat{\mathbf{P}}_{K,K-1}^T \\ \ddots & \ddots & \ddots & \ddots & \hat{\mathbf{P}}_{K,K-1} & \hat{\mathbf{P}}_K \end{bmatrix}, \quad (\text{A.19})$$

where we note the matrix is in general dense but we have only assigned symbols to the blocks we will use. Multiplying out $\mathbf{L}^{-T}\mathbf{L}^{-1}$ and comparing to $\hat{\mathbf{P}}$ we can establish a backward recursive relationship:

$$\hat{\mathbf{P}}_{k-1} = \mathbf{L}_{k-1}^{-T} \left(\mathbf{1} + \mathbf{L}_{k,k-1}^T \hat{\mathbf{P}}_k \mathbf{L}_{k,k-1} \right) \mathbf{L}_{k-1}^{-1}, \quad (\text{A.20a})$$

$$\hat{\mathbf{P}}_{k,k-1} = -\hat{\mathbf{P}}_k \mathbf{L}_{k,k-1} \mathbf{L}_{k-1}^{-1}, \quad (\text{A.20b})$$

which we initialize with

$$\hat{\mathbf{P}}_K = \mathbf{L}_K^{-T} \mathbf{L}_K^{-1}. \quad (\text{A.21})$$

As we have already computed all of the blocks of \mathbf{L} in the Cholesky smoother, we can simply include this calculation in the backward pass if we want the posterior covariance associated with our estimate. This does not change the complexity of the overall algorithm, which remains at $O(K)$, albeit with a slightly higher coefficient.

A.3.2 Posterior Covariance in the RTS Smoother

We can manipulate the covariance backward recursion into the canonical RTS form as follows. First, we note that

$$\begin{aligned} \mathbf{L}_{k,k-1}\mathbf{L}_{k-1}^{-1} &= \underbrace{\mathbf{L}_{k,k-1}\mathbf{L}_{k-1}^T}_{-\mathbf{Q}_k^{-1}\mathbf{A}_{k-1}} \quad \underbrace{\mathbf{L}_{k-1}^{-T}\mathbf{L}_{k-1}^{-1}}_{(\mathbf{I}_{k-1} + \mathbf{A}_{k-1}^T\mathbf{Q}_k^{-1}\mathbf{A}_{k-1})^{-1}} \\ &= -\underbrace{(\mathbf{A}_{k-1}\hat{\mathbf{P}}_{k-1,f}\mathbf{A}_{k-1}^T + \mathbf{Q}_k)}_{\check{\mathbf{P}}_{k,f}}^{-1}\mathbf{A}_{k-1}\hat{\mathbf{P}}_{k-1,f} \\ &= -\left(\hat{\mathbf{P}}_{k-1,f}\mathbf{A}_{k-1}^T\check{\mathbf{P}}_{k,f}^{-1}\right)^T. \quad (\text{A.22}) \end{aligned}$$

Also, we have that

$$\begin{aligned} \mathbf{L}_{k-1}^{-T}\mathbf{L}_{k-1}^{-1} &= (\mathbf{I}_{k-1} + \mathbf{A}_{k-1}^T\mathbf{Q}_k^{-1}\mathbf{A}_{k-1})^{-1} \\ &= \hat{\mathbf{P}}_{k-1,f} - \hat{\mathbf{P}}_{k-1,f}\mathbf{A}_{k-1}^T\underbrace{(\mathbf{A}_{k-1}\hat{\mathbf{P}}_{k-1,f}\mathbf{A}_{k-1}^T + \mathbf{Q}_k)}_{\check{\mathbf{P}}_{k,f}}^{-1}\mathbf{A}_{k-1}\hat{\mathbf{P}}_{k-1,f} \\ &= \hat{\mathbf{P}}_{k-1,f} - \left(\hat{\mathbf{P}}_{k-1,f}\mathbf{A}_{k-1}^T\check{\mathbf{P}}_{k,f}^{-1}\right)\check{\mathbf{P}}_{k,f}\left(\hat{\mathbf{P}}_{k-1,f}\mathbf{A}_{k-1}^T\check{\mathbf{P}}_{k,f}^{-1}\right)^T. \quad (\text{A.23}) \end{aligned}$$

Plugging these two results into (A.20a) we have

$$\hat{\mathbf{P}}_{k-1} = \hat{\mathbf{P}}_{k-1,f} + \left(\hat{\mathbf{P}}_{k-1,f}\mathbf{A}_{k-1}^T\check{\mathbf{P}}_{k,f}^{-1}\right)\left(\hat{\mathbf{P}}_k - \check{\mathbf{P}}_{k,f}\right)\left(\hat{\mathbf{P}}_{k-1,f}\mathbf{A}_{k-1}^T\check{\mathbf{P}}_{k,f}^{-1}\right)^T, \quad (\text{A.24})$$

which we initialize with $\hat{\mathbf{P}}_K = \hat{\mathbf{P}}_{K,f}$ and iterate backward. Finally, we can also plug the same two results into (A.20b) to obtain

$$\hat{\mathbf{P}}_{k,k-1} = \hat{\mathbf{P}}_k \left(\hat{\mathbf{P}}_{k-1,f}\mathbf{A}_{k-1}^T\check{\mathbf{P}}_{k,f}^{-1} \right)^T, \quad (\text{A.25})$$

for the blocks above and below the main diagonal of the full covariance matrix, $\hat{\mathbf{P}}$; these are needed, for example, when interpolating for additional times of interest using (3.198b).