Information Science and Statistics

Paul Fieguth

Statistical Image Processing and Multidimensional Modeling

Springer

Statistical Image Processing and Multidimensional Modeling

Information Science and Statistics

For other titles published in this series, go to
www.springer.com/series/3816

Paul Fieguth

Statistical Image Processing and Multidimensional Modeling

Springer

Paul Fieguth
Department of Systems Design Engineering
Faculty of Engineering
University of Waterloo
Waterloo Ontario N2L-3G1
Canada
pfieguth@uwaterloo.ca

ISSN 1613-9011
ISBN 978-1-4419-7293-4 e-ISBN 978-1-4419-7294-1
DOI 10.1007/978-1-4419-7294-1
Springer New York Dordrecht Heidelberg London

Library of Congress Control Number: 2010938436

Preface

As a young professor in 1997 I taught my graduate course in *Stochastic Image Processing* for the first time. Looking back on my rough notes from that time, the course must have been a near impenetrable disaster for the graduate students enrolled, with a long list of errors, confusions, and bad notation.

With every repetition the course improved, with significant changes to notation, content, and flow. However, at the same time that a cohesive, large-scale form of the course took shape, the absence of any textbook covering this material became increasingly apparent. There are countless texts on the subjects of image processing, Kalman filtering, and signal processing, however precious little for random fields or spatial statistics. The few texts that *do* cover Gibbs models or Markov random fields tend to be highly mathematical research monographs, not well suited as a textbook for a graduate course.

More than just a graduate course textbook, this text was developed with the goal of being a useful reference for graduate students working in the areas of image processing, spatial statistics, and random fields. In particular, there are many concepts which are known and documented in the research literature, which are useful for students to understand, but which do not appear in many textbooks. This perception is driven by my own experience as a PhD student, which would have been considerably simplified if I had had a text accessible to me addressing some of the following gaps:

- FFT-based estimation (Section 8.3)
- A nice, simple, clear description of multigrid (Section 9.2.5)
- The inference of dynamic models from cross-statistics (Chapter 10)
- A clear distinction and relationship between squared and unsquared kernels (Chapter 5)

- A graphical summary relating Gibbs and Markov models (Figure 6.11)

To facilitate the use of this textbook and the methods described within it, I am making available online (see page XV) much of the code which I developed for this text. This code, some colour figures, and (hopefully few) errata can be found from this book's home page:

http://ocho.uwaterloo.ca/book

This text has benefited from the work, support, and ideas of a great many people. I owe a debt of gratitude to the countless researchers upon whose work this book is built, and who are listed in the bibliography. Please accept my apologies for any omissions.

The contents of this book are closely aligned with my research interests over the past ten years. Consequently the work of a number of my former graduate students appears in some form in this book, and I would like to recognize the contributions of Simon Alexander, Wesley Campaigne, Gabriel Carballo, Michale Jamieson, Fu Jin, Fakhry Khellah, Ying Liu, and Azadeh Mohebi.

I would like to thank my Springer editor, John Kimmel, who was an enthusiastic supporter of this text, and highly tolerant of my slow pace in writing. Thanks also to copy editor Valerie Greco for her careful examination of grammar and punctuation (and where any remaining errors are mine, not hers). I would like to thank the anonymous reviewers, who read the text thoroughly and who provided exceptionally helpful constructive criticism. I would also like to thank the *non*-anonymous reviewers, friends and students who gave the text another look: Werner Fieguth, Betty Pries, Akshaya Mishra, Alexander Wong, Li Liu, and Gerald Mwangi.

I would like to thank Christoph Garbe and Michael Winckler, the two people who coordinated my stay at the University of Heidelberg, where this text was completed. My thanks to the Deutscher Akademischer Austausch Dienst, the Heidelberg Graduate School, and to the Heidelberg Collaboratory for Image Processing for supporting my visit.

Many thanks and appreciation to Betty for encouraging this project, and to the kids in Appendix C for just being who they are.

Waterloo, Ontario

Paul Fieguth
July, 2010

Contents

Preface **V**

Table of Contents **VII**

List of Examples **XIII**

List of Code Samples **XV**

Nomenclature **XVII**

1 Introduction **1**

Part I Inverse Problems and Estimation **11**

2 Inverse Problems **13**
2.1 Data Fusion 16
2.2 Posedness 19
2.3 Conditioning 23
2.4 Regularization and Prior Models 29
2.4.1 Deterministic Regularization 34
2.4.2 Bayesian Regularization 37
2.5 Statistical Operations 40
2.5.1 Canonical Problems 40
2.5.2 Prior Sampling 42
2.5.3 Estimation 44
2.5.4 Posterior Sampling 49
2.5.5 Parameter Estimation 50
Application 2: Ocean Acoustic Tomography 50
Summary 52
For Further Study 53
Sample Problems 53

3 Static Estimation and Sampling **57**
3.1 Non-Bayesian Estimation 58

3.2 Bayesian Estimation 64
3.2.1 Bayesian Static Problem 67
3.2.2 Bayesian Estimation and Prior Means 68
3.2.3 Approximate Bayesian Estimators 70
3.2.4 Bayesian / NonBayesian Duality 73
3.3 Static Sampling 74
3.4 Data Fusion 76
Application 3: Atmospheric Temperature Inversion [282] 78
Summary 80
For Further Study 81
Sample Problems 81

4 Dynamic Estimation and Sampling 85
4.1 The Dynamic Problem 86
4.1.1 First-Order Gauss–Markov Processes 88
4.1.2 Static — Dynamic Duality 89
4.2 Kalman Filter Derivation 93
4.3 Kalman Filter Variations 100
4.3.1 Kalman Filter Algorithms 102
4.3.2 Steady-State Kalman Filtering 107
4.3.3 Kalman Filter Smoother 109
4.3.4 Nonlinear Kalman Filtering 114
4.4 Dynamic Sampling 118
4.5 Dynamic Estimation for Discrete-State Systems 119
4.5.1 Markov Chains 119
4.5.2 The Viterbi Algorithm 120
4.5.3 Comparison to Kalman Filter 122
Application 4: Temporal Interpolation of Ocean Temperature [191] 125
Summary 125
For Further Study 127
Sample Problems 127

Part II Modelling of Random Fields 131

5 Multidimensional Modelling 133
5.1 Challenges 134
5.2 Coupling and Dimensionality Reduction 135
5.3 Sparse Storage and Computation 139
5.3.1 Sparse Matrices 139
5.3.2 Matrix Kernels 141
5.3.3 Computation 143
5.4 Modelling 148
5.5 Deterministic Models 149
5.5.1 Boundary Effects 153

5.5.2 Discontinuity Features . . . 155
5.5.3 Prior-Mean Constraints . . . 156
5.6 Statistical Models . . . 158
5.6.1 Analytical Forms . . . 160
5.6.2 Analytical Forms and Nonstationary Fields . . . 164
5.6.3 Recursive / Dynamic Models . . . 166
5.6.4 Banded Inverse-Covariances . . . 167
5.7 Model Determination . . . 169
5.8 Choice of Representation . . . 172
Application 5: Synthetic Aperture Radar Interferometry [53] . . . 173
For Further Study . . . 175
Sample Problems . . . 175

6 Markov Random Fields **179**
6.1 One-Dimensional Markovianity . . . 180
6.1.1 Markov Chains . . . 181
6.1.2 Gauss–Markov Processes . . . 181
6.2 Multidimensional Markovianity . . . 182
6.3 Gauss–Markov Random Fields . . . 185
6.4 Causal Gauss–Markov Random Fields . . . 189
6.5 Gibbs Random Fields . . . 192
6.6 Model Determination . . . 199
6.6.1 Autoregressive Model Learning . . . 199
6.6.2 Noncausal Markov Model Learning . . . 201
6.7 Choices of Representation . . . 207
Application 6: Texture Classification . . . 208
Summary . . . 211
For Further Study . . . 212
Sample Problems . . . 212

7 Hidden Markov Models **215**
7.1 Hidden Markov Models . . . 216
7.1.1 Image Denoising . . . 216
7.1.2 Image Segmentation . . . 219
7.1.3 Texture Segmentation . . . 220
7.1.4 Edge Detection . . . 221
7.2 Classes of Joint Markov Models . . . 222
7.3 Conditional Random Fields . . . 225
7.4 Discrete-State Models . . . 227
7.4.1 Local Gibbs Models . . . 228
7.4.2 Nonlocal Statistical-Target Models . . . 229
7.4.3 Local Joint Models . . . 230
7.5 Model Determination . . . 231
Application 7: Image Segmentation . . . 233
For Further Study . . . 237

Sample Problems 237

8 Changes of Basis **241**
8.1 Change of Basis 243
8.2 Reduction of Basis 247
8.2.1 Principal Components 248
8.2.2 Multidimensional Basis Reduction 253
8.2.3 Local Processing 259
8.3 FFT Methods 262
8.3.1 FFT Diagonalization 263
8.3.2 FFT and Spatial Models 265
8.3.3 FFT Sampling and Estimation 266
8.4 Hierarchical Bases and Preconditioners 269
8.4.1 Interpolated Hierarchical Bases 272
8.4.2 Wavelet Hierarchical Bases 273
8.4.3 Wavelets and Statistics 275
8.5 Basis Changes and Markov Random Fields 278
8.6 Basis Changes and Discrete-State Fields 281
Application 8: Global Data Assimilation [111] 285
Summary 287
For Further Study 288
Sample Problems 288

Part III Methods and Algorithms **291**

9 Linear Systems Estimation **293**
9.1 Direct Solution 295
9.1.1 Gaussian Elimination 295
9.1.2 Cholesky Decomposition 295
9.1.3 Nested Dissection 296
9.2 Iterative Solution 298
9.2.1 Gauss–Jacobi / Gauss–Seidel 299
9.2.2 Successive Overrelaxation (SOR) 303
9.2.3 Conjugate Gradient and Krylov Methods 306
9.2.4 Iterative Preconditioning 310
9.2.5 Multigrid 313
Application 9: Surface Reconstruction 320
For Further Study 321
Sample Problems 322

10 Kalman Filtering and Domain Decomposition **325**
10.1 Marching Methods 327
10.2 Efficient, Large-State Kalman Filters 330
10.2.1 Large-State Kalman Smoother 331

10.2.2 Steady-State KF . . . 334
10.2.3 Strip KF . . . 334
10.2.4 Reduced-Update KF . . . 336
10.2.5 Sparse KF . . . 337
10.2.6 Reduced-Order KF . . . 338
10.3 Multiscale . . . 339
Application 10: Video Denoising [178] . . . 347
Summary . . . 350
For Further Study . . . 351
Sample Problems . . . 351

11 Sampling and Monte Carlo Methods 355
11.1 Dynamic Sampling . . . 357
11.2 Static Sampling . . . 358
11.2.1 FFT . . . 361
11.2.2 Marching . . . 362
11.2.3 Multiscale Sampling . . . 363
11.3 MCMC . . . 365
11.3.1 Stochastic Sampling . . . 366
11.3.2 Continuous-State Sampling . . . 369
11.3.3 Large-Scale Discrete-State Sampling . . . 370
11.4 Nonparametric Sampling . . . 374
Application 11: Multi-Instrument Fusion of Porous Media [236] . . . 377
For Further Study . . . 379
Sample Problems . . . 379

Part IV Appendices 381

A Algebra 383
A.1 Linear Algebra . . . 383
A.2 Matrix Operations . . . 385
A.3 Matrix Positivity . . . 388
A.4 Matrix Positivity of Covariances . . . 389
A.5 Matrix Types . . . 391
A.6 Matrix / Vector Derivatives . . . 391
A.7 Matrix Transformations . . . 395
A.7.1 Eigendecompositions . . . 396
A.7.2 Singular Value Decomposition . . . 400
A.7.3 Cholesky, Gauss, LU, Gram–Schmidt, QR, Schur . . . 401
A.8 Matrix Square Roots . . . 407
A.9 Pseudoinverses . . . 409

B Statistics 411
B.1 Random Variables, Random Vectors, and Random Fields . . . 411

B.1.1 Random Variables 411
B.1.2 Joint Statistics 412
B.1.3 Random Vectors 414
B.1.4 Random Fields 415
B.2 Transformation of Random Vectors 416
B.3 Multivariate Gaussian Distribution 417
B.4 Covariance Matrices 419

C Image Processing **423**
C.1 Convolution 424
C.2 Image Transforms 428
C.3 Image Operations 430

Reference Summary **433**

References **437**

Index **451**

List of Examples

Application 2	Ocean Acoustic Tomography	50
Application 3	Atmospheric Temperature Inversion [282]	78
Application 4	Temporal Interpolation of Ocean Temperature [191]	122
Application 5	Synthetic Aperture Radar Interferometry [53]	173
Application 6	Texture Classification	208
Application 7	Image Segmentation	232
Application 8	Global Data Assimilation [111]	285
Application 9	Surface Reconstruction	318
Application 10	Video Denoising [178]	345
Application 11	Multi-Instrument Fusion of Porous Media [236]	377

Example 2.1	Root Finding is an Inverse Problem	16
Example 2.2	Interpolation and Posedness	20
Example 2.3	Regression and Posedness	23
Example 2.4	Measurement Models and Conditioning	25
Example 2.5	Matrix Conditioning	27
Example 2.6	Interpolation and Regularization	32
Example 2.7	Interpolation and Smoothness Models	36
Example 2.8	Interpolation and Cross Validation	38
Example 2.9	State Estimation and Sampling	46
Example 3.1	Linear Regression is Least Squares	62
Example 3.2	Simple Scalar Estimation	69
Example 3.3	Prior Mean Removal	71
Example 3.4	Static Estimation and Sampling	75
Example 4.1	Dynamic Estimation and Sampling	92
Example 4.2	Kalman Filtering and Interpolation	98
Example 4.3	Recursive Smoothing and Interpolation	112
Example 5.1	Multidimensional Interpolation	159
Example 5.2	Model Inference	170
Example 6.1	Example Markov Kernels	190
Example 6.2	Robust Estimation	198
Example 6.3	Markov Model Inference and Model Order	204
Example 6.4	Model Inference and Conditioning	206
Example 7.1	Multiple Hidden Fields	223

Example 8.1 Principal Components for Remote Sensing 252
Example 8.2 Overlapped Local Processing 261
Example 8.3 FFT and Spatial Statistics 268
Example 8.4 Hierarchical Bases 276
Example 9.1 Iterative Solutions to Interpolation 302
Example 9.2 Eigendistributions of Iterative Interpolators 305
Example 10.1 Interpolation by Marching 332
Example 11.1 Annealing and Energy Barriers 372

Algorithm 1 The Kalman Filter 97
Algorithm 2 Simplified Expectation Maximization 232
Algorithm 3 FFT Estimation and Sampling 267
Algorithm 4 The Cholesky Decomposition 296
Algorithm 5 Gauss–Seidel 300
Algorithm 6 Conjugate Gradient 309
Algorithm 7 Multigrid 315
Algorithm 8 MATLAB Multigrid implementation I 318
Algorithm 9 MATLAB Multigrid implementation II 319
Algorithm 10 Single Gibbs Sample 366
Algorithm 11 Single Metropolis Sample 367
Algorithm 12 Basic, Single-Scale Annealing 369
Algorithm 13 Multilevel Annealing, Initializing from Coarser Scale 373
Algorithm 14 Multilevel Annealing, Constrained by Coarser Scale 374

List of Code Samples

To make the methods and algorithms described in this book as useful and accessible as possible, a variety of MATLAB scripts, written as part of the development of this textbook, are being made available.

The intent of the scripts is certainly not to be directly applicable to large multidimensional problems. Instead, the code is provided to give the reader some simple examples which actually work, which illustrate the concepts documented in the text, and which hopefully more rapidly lead the reader to be able to do implementations of his or her own.

The code can be found from the text website at

http://ocho.uwaterloo.ca/book

Example 2.4: cond_num.m 25
Example 2.5: cond_num.m 28
Example 2.7: interp_reg.m 36
Example 2.8: cross_val.m 38
Figure 5.7: bands.m 145
Figure 5.8: kernels.m 152
Figure 5.15: corr_len.m 158
Example 5.1: twod_interp.m 159
Figure 5.16: posdef.m 161
Figure 5.18: bands.m 168
Figure 6.2: mrf.m 182
Example 6.1: mrf.m 190
Example 6.2: robust.m 198
Example 6.3: mrf.m 204
Example 6.3: mrf.m 207
Figure 8.2: iterative.m 243
Figure 8.7: basisreduc.m 255
Figure 8.8: basisreduc.m 258
Example 8.2: ovlp_demo.m 261
Example 8.3: fft_mrf.m 268
Example 8.4: precond_hier.m 276

Figure 8.17: mrf.m 280
Example 9.2: iterative.m 302
Example 9.2: iterative.m 305
Figure 9.10: multigrid.m 316
Example 10.1: marching.m 332
Figure 11.3: twod_sample.m 359
Figure 11.3: twod_sample.m 361
Figure 11.4: fft_discrete.m 362
Figure A.1: posdef.m 390
Figure B.3: pdf.m 421

Nomenclature

The following tables of nomenclature are designed to assist the reader in understanding the mathematical language used throughout this text. In the author's opinion this is of considerable value particularly for readers who seek to use the book as a reference and need to be able to understand individual equations or sections without reading an entire chapter for context. Four sets of definitions follow:

1. Basic syntax
2. Mathematical functions
3. Definitions of commonly-used variables
4. Notation for spatial models

Page references are given to provide a few examples of use and some context to the notation, but are in no way intended to be exhaustive.

We limit ourselves here to just defining the notation. For an explanation of algebraic concepts (matrix transpose, eigendecomposition, inverse, etc.) the reader is referred to Appendix A. For an explanation of related statistical concepts (expectation, covariance, etc.), see Appendix B. A brief overview of image processing can be found in Appendix C. Most of the spatial models are explained in Chapters 5 and 6.

Syntax	Definition	Page References
a	scalar, random variable	16 411
$\underline{a}$	column vector, random vector	13 414
a_i	ith element of vector $\underline{a}$	16 414
$\underline{a}_i$	ith vector in a sequence	15 294
a_{ij}	i,jth element of matrix A	299 385
A	matrix	13 383
A^T	matrix transpose	31 385
A^H	matrix Hermitian (complex transpose)	392
A^{-1}	matrix inverse	20 386
$\|A\|$	matrix determinant	63 386 418
$\mathcal{A}$	kernel corresponding to stationary matrix A	142 143 151
$\mathcal{A}^{-1}$	kernel corresponding to A^{-1}, but $\mathcal{A}^{-1} \neq (\mathcal{A})^{-1}$	146
$\mathcal{A}^T$	kernel corresponding to A^T, but $\mathcal{A}^T \neq (\mathcal{A})^T$	146 152 166
$\mathbb{R}^n$	real vector of length n	19 253 384
$\mathbb{R}^{k\times n}$	real $k \times n$ array	143 383
$[A]$, $[A]_{n\times 1}$	reordering of matrix to column vector	133 141 166
$[\underline{a}]_{n\times m}$	reordering of column vector to $n \times m$ matrix	133 146
$[\underline{a}]_{\underline{n}}$	reordering to $n_1 \times n_2 \times \ldots$ multidimensional array	265
$\hat{a}, \hat{\underline{a}}, \hat{A}$	estimate of $a, \underline{a}, A$	22 58
$\tilde{\underline{a}}$	estimation error in $\hat{\underline{a}}$	64 108
$\bar{a}, \bar{\underline{a}}, \bar{A}$	transformation of $a, \underline{a}, A$	41 241
$\check{\underline{a}}$	given sample data of $\underline{a}$	201 412
$\tilde{P}$	estimation error covariance	68 72 294
$\Pr(Q)$	probability of some event Q	119 181 411
$p(x)$	probability density of x	42 65 411
$p(x\|y)$	conditional probability density	45 179 413
$\{\ldots\}$	a set	15 415
$\|S\|$	number of elements in set S	15 121 203
$\|\underline{x}\|, \|A\|$	vector norm for $\underline{x}$, matrix norm for A	22 59
$\|\underline{x}\|_P = \underline{x}^T P \underline{x}$	vector squared-norm for $\underline{x}$ with respect to covariance P	31 63 73
$\textcircled{a}$	convolution kernel origin	145 152 268
$\sim$	is distributed as ...	49 355
$\underline{x} \sim P$	$\underline{x}$ has covariance P; the mean is zero or not of interest	28 141 241
$\underline{x} \sim (\underline{\mu}, P)$	$\underline{x}$ has mean $\underline{\mu}$ and covariance P, distribution unknown	37 69 74
$\underline{x} \sim \mathcal{N}(\underline{\mu}, P)$	$\underline{x}$ is Gaussian with mean $\underline{\mu}$ and covariance P	28 63 417

Tab. Notation.1. Basic vector, matrix, and statistical syntax

Function	Definition	Page References
$\text{sign}(a)$	the sign of scalar a, $\text{sign}(a) = \begin{cases} -1 & a < 0 \\ 0 & a = 0 \\ 1 & a > 0 \end{cases}$	144 400
$a \bmod b$	division modulus (remainder)	155 262 392
$\min(\cdot)$	the minimum value in a set	198 385 385
$\min_{x \in X}(\cdot)$	the minimum value of a function over range X	401
$\arg_x \min(\cdot)$	the value of x which minimizes the function	30 63 409
$\text{Ra}(A)$	the range space of A	19 53 385
$\text{Nu}(A)$	the null space of A	19 53 384
$\text{rank}(A)$	the rank of A	53 385
$\dim(\cdot)$	the dimension of a space	384
$\text{tr}(A)$	the trace, the sum of the diagonal elements of A	247 387 395
$\det(A)$	the determinant of A	24 386 387
$\kappa(A)$	matrix condition number of A	26 104 246
$\text{diag}(A)$	a vector containing the diagonal elements of A	54 140 266
$\text{Diag}(\underline{x})$	a diagonal matrix with $\underline{x}$ along the diagonal	165 264 294
I	the identity matrix	37 277 357
FFT_d	the d-dimensional fast Fourier transform	267 361 428
FFT_d^{-1}	the d-dimensional inverse fast Fourier transform	265 361
WT	the wavelet transform	273 347 428
WT^{-1}	the inverse wavelet transform	275 290
$\odot$	element-by-element matrix multiplication	146 165 266
$\oslash$	element-by-element matrix division	146 254 265
$*$	convolution	142 146 424
$\circledast$	circular convolution	427 428
$\text{var}(\cdot)$	variance	164 389
$\text{cov}(\cdot)$	covariance	42 67 87
$E[\cdot]$	expectation	42 64 412
$E_a[\cdot]$	expectation over variable a, if otherwise ambiguous	232 363
$\equiv$	is equivalent to, identical to	31 61
$\triangleq$	is defined as	15 58
$< > \geq \leq$	inequalities, in positive-definite sense for matrices	81 100 388

Tab. Notation.2. Mathematical functions and operations (see Appendix A)

Symbol	Definition	Page References
b	linear system target	245 293 403
b	estimator bias	65 66
c	random field clique	193 368
d	spatial dimensionality	262 267
e	error	54 58 298
$\underline{e}_i$	the ith unit vector: all zeros with a one in the ith position	384
f	forward problem	13 15 30
g	Markov random field model coefficient	186 202
i, j	general indices	17 37
k, n, q	matrix and vector dimensions	19 140 148
m	measurement	13 40 58
p	probability density	42 65 411
r	linear system residual	298 306 314
s, t	time	42 86
v	measurement noise	13 40 58
v	eigenvector	249 304 396
w	dynamic process noise	42 86 325
x, y	spatial location or indices	35 150 221
z	system state	13 40 58
A	linear system, normal equations	245 293 403
A	dynamic model predictor	86 143 325
B	dynamic model stochastic weight	42 86 325
C	measurement model	13 40 58
E	expectation	42 64 412
F	Fourier transform	257 263
F	change of basis (forwards)	241 247 314
G	Markov random field model	186 201
H	energy function	192 222 371
I	identity matrix	37 277 357
J	optimization criterion	198 249
K	estimator gain	97 108 330
L	constraints matrix	150 157 293
M	measurements field (multidimensional)	170 215 267
N	image or patch size	134 214 329
P	state covariance	40 143 160
Q	squared system constraints	152 152
R	measurement noise covariance	40 63 87
S	change of basis (backwards)	246 247 314
T	annealing temperature	192 368
U, V	orthogonal matrices	245 250 400
W	wavelet transform	277 348
Z	random field (multidimensional)	133 141 327

Tab. Notation.3. Symbol definitions

Symbol	Definition	Page References		
α, β, γ	constants	41	98	164
β	Gibbs inverse temperature	192	228	355
δ	Dirac delta	87	197	200
δ	small offset or perturbation	26	160	184
ϵ	small amount		21	305
κ	matrix condition number	26	104	246
θ	model parameters		50	170
λ	regularization parameter	31	35	63
λ	eigenvalue	304	396	420
μ	mean	37	67	412
ν	estimator innovations	95	96	110
ρ	correlation coefficient		390	413
ρ	spectral radius		300	398
σ	standard deviation	69	102	412
σ	singular value	27	256	400
τ	time offset or period		44	111
ξ	correlation length		28	124
ζ	threshold	198	300	432
Γ	covariance square root	104	166	401
Λ, Σ	covariance	66	93	389
Ψ	state space or alphabet	119	122	181
Ω	problem space (multidimensional lattice)	184	193	415
Ξ	region subset operator		167	342
$\mathcal{B}$	matrix banding structure	144	146	167
$\mathcal{C}$	clique set		193	194
$\mathcal{N}$	neighbourhood	184	189	226
$\mathcal{N}(\mu, P)$	Gaussian distribution with mean $\underline{\mu}$, covariance P	28	63	417
$\mathcal{O}(\cdot)$	complexity order		143	326
$\mathbb{R}$	real			374
$\mathbb{R}^n$	real vector of length n	19	253	384
$\mathbb{R}^{k \times n}$	real $k \times n$ array		143	383
$\mathbb{Z}$	Gibbs partition function		192	355
$0,\ 1$	scalar constant zero, one		24	59
$\underline{0},\ \underline{1}$	vector constants of all zeros, all ones	19	45	72
$\mathbf{0},\ \mathbf{1}$	matrix constants of all zeros, all ones		66	76

Tab. Notation.3. Symbol definitions (cont'd)

Nonstationary Model (Dense)	Stationary Model (Kernel)	Model Type	Page References		
A, B	$\mathcal{A}, \mathcal{B}$	Square root model (dynamic)	86	143	325
Γ	$\mathbf{\Gamma}$	Square root model (static)	104	166	401
P	$\mathcal{P}$	Squared model (covariance)	40	143	160
V	$\mathcal{V}$	Square root inverse model (Gibbs field)			193
G	$\mathcal{G}$	Squared inverse model (Markov field)		186	201
L	$\mathcal{L}$	Square root deterministic model (constraints)	150	157	293
Q	$\mathcal{Q}$	Squared deterministic model		152	152
C, R	$\mathcal{C}, \mathcal{R}$	Measurement model	13	40	58

Tab. Notation.4. Spatial model definitions

1

Introduction

Images are all around us! Inexpensive digital cameras, video cameras, computer web-cams, satellite imagery, and images off the Internet give us access to spatial imagery of all sorts. The vast majority of these images will be of scenes at human scales — pictures of animals / houses / people / faces and so on — relatively complex images which are not well described statistically or mathematically. Many algorithms have been developed to process / denoise / compress / segment such images, described in innumerable textbooks on image processing [36, 54, 143, 174, 210], and briefly reviewed in Appendix C.

Somewhat less common, but of great research interest, are images which do allow some sort of mathematical characterization, and to which standard image-processing algorithms may not apply. In most cases we do not necessarily have *images* here, per se, but rather spatial datasets, with one or more measurements taken over a two- or higher-dimensional space.

There are many important problems falling into this latter group of scientific images, and where this text seeks to make a contribution. Examples abound throughout remote sensing (satellite data mapping, data assimilation, sea-ice / climate-change studies, land use), medical imaging (denoising, organ segmentation, anomaly detection), computer vision (textures, image classification, segmentation), and other 2D / 3D problems (groundwater, biological imaging, porous media, etc.).

Although a great deal of research has been applied to scientific images, in most cases the resulting methods are not well documented in common textbooks, such that many experienced researchers will be unfamiliar with the use of the FFT method (Section 8.3) or of posterior sampling (Chapter 11), for example.

The goal, then, of this text is to address methods for solving multidimensional inverse problems. In particular, the text seeks to avoid the pitfall of being entirely mathematical / theoretical at one extreme, or primarily applied / algorithmic on the other, by deliberately developing the basic theory (Part I), the mathematical mod-

P. Fieguth, *Statistical Image Processing and Multidimensional Modeling*, Information Science and Statistics, DOI 10.1007/978-1-4419-7294-1_1, © Springer Science+Business Media, LLC 2011

elling (Part II), and the algorithmic / numerical methods (Part III) of solving a given problem.

Inverse Problems

So, to begin, why would we want to solve an inverse problem?

There are a great many spatial phenomena that a person might want to study ...

- The salinity of the ocean surface as a function of position;
- The temperature of the atmosphere as a function of position;
- The height of the grass growing in your back yard, as a function of location;
- The proportions of oil and water in an oil reservoir.

In each of these situations, you aren't just handed a map of the spatial process you wish to study, rather you have to *infer* such a map from given measurements. These measurements might be a simple function of the spatial process (such as measuring the height of the grass using a ruler) or might be complicated nonlinear functions (such as microwave spectra for inferring temperature).

The process by which measurements are generated from the spatial process is normally relatively straightforward, and is referred to as a *forward problem*. More difficult, then, is the *inverse problem*, discussed in detail in Chapter 2, which represents the mathematical inverting of the *forward problem*, allowing you to infer the process of interest from the measurements. A simple illustration is shown in Figure 1.1.

Large Multidimensional Problems

So why is it that we wish to study large multidimensional problems?

The solution to linear inverse problems (see Chapter 3) is easily formulated analytically, and even a nonlinear inverse problem can be reformulated as an optimization problem and solved. The challenge, then, is not the solving of inverse problems *in principle*, but rather actually solving them *in practice*.

For example, the solution to a linear inverse problem involves a matrix inversion. As the problem is made larger and larger, eventually the matrix becomes computationally or numerically impossible to invert. However, this is not just an abstract limit — even a modest two-dimensional problem at a resolution of 1000×1000 pixels contains one million unknowns, which would require the inversion of a one-million by one-million matrix: completely unfeasible.

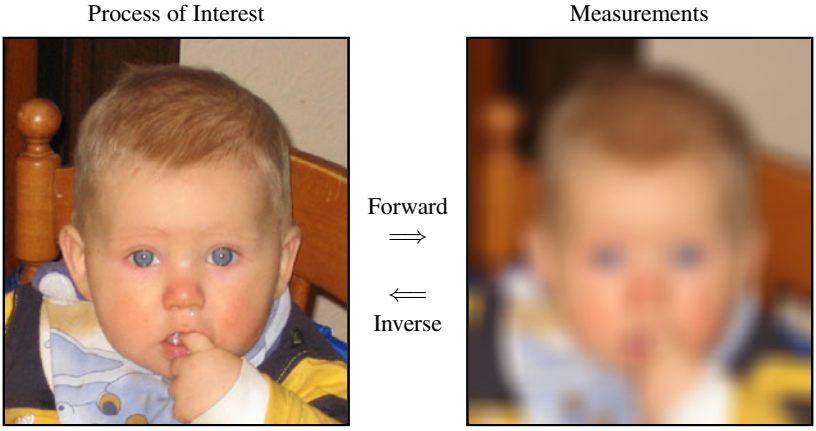


Fig. 1.1. An inverse problem: You want a nice clear photo of a face, however your camera yields blurry measurements. To solve this inverse problem requires us to mathematically invert the forward process of blurring.

Therefore even rather modestly sized two- and higher-dimensional problems become impossible to solve using straightforward techniques, yet these problems are very common. Problems having one million or more unknowns are littered throughout the fields of remote sensing, oceanography, medical imaging, and seismology, to name a few.

To be clear, a problem is considered to be multidimensional if it is a function of two or more independent variables. These variables could be spatial (as in a two-dimensional image or a three-dimensional volume), spatio-temporal (such as a video, a sequence of two-dimensional images over time), or a function of other variables under our control.

Multidimensional Methods versus Image Processing

What is it that the great diversity of algorithms in the image processing literature cannot solve?

The majority of images which are examined and processed in image processing are "real" images, pictures and scenes at human scales, where the images are not well described mathematically. Therefore the focus of image processing is on making relatively few explicit, mathematical assumptions about the image, and instead focusing on the development of algorithms that perform image-related tasks (such as compression, segmentation, edge detection, etc.).

Face Grass Clouds

Fig. 1.2. Which of these might be best characterized mathematically? Many natural phenomena, when viewed at an appropriate scale, have a behaviour which is sufficiently varied or irregular that it can be modelled via relatively simple equations, as opposed to a human face, which would need a rather complex model to be represented accurately.

In contrast, of great research interest are images taken at microscopic scales (cells in a Petri dish, the crystal structure of stone or metal) or at macroscopic scales (the temperature distribution of the ocean or of the atmosphere, satellite imagery of the earth) which do, in general, allow some sort of mathematical characterization, as explored in Figure 1.2. That is, the focus of this text is on the assumption or inference of rather *explicit* mathematical models of the unknown process.

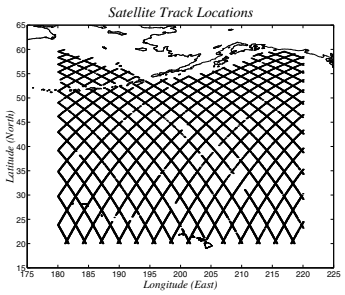
Next, in order to be able to say something about a problem, we need measurements of it. These measurements normally suffer from one of three issues, any one of which would preclude the use of standard image-processing techniques:

1. For measurements produced by a scientific instrument, acquiring a measurement normally requires time and/or money, therefore the number of measurements is constrained. Frequently this implies that the multidimensional problem of interest is only sparsely sampled, as illustrated in Figure 1.3.

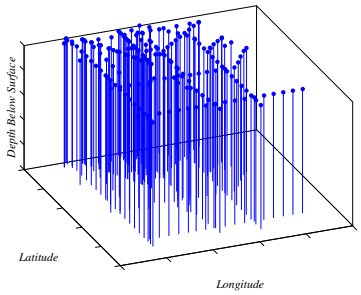
   There exist many standard methods to interpolate gaps in a sequence of data, however standard interpolation knows nothing about the underlying phenomenon being studied. That is, surely a grass-like texture should be interpolated differently from a map of ocean-surface temperature.

2. Most measurements are not exact, but suffer from some degree of noise. Ideally we would like to remove this noise, to infer a more precise version of the underlying multidimensional phenomenon.

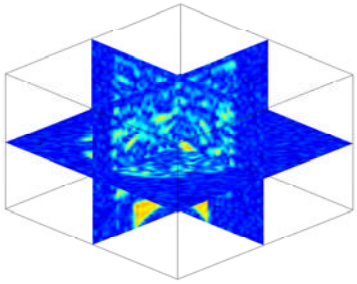
   There exist many algorithms for noise reduction in images, however these are necessarily heuristic, because they are designed to work on photographic images, which might contain images of faces / cars / trees and the like. Given a scientific dataset, surely we would wish to undertake denoising in a more systematic (ideally optimal) manner, somehow dependent on the behaviour of the underlying phenomenon.



Satellite Altimetry
(each point measures ocean height)



Ship-Based Measurements
(each point measures ocean temperature)



Slices from a 3D MRI
(each point represents the concentration of water in a block of concrete)

Fig. 1.3. Multidimensional measurements: Three examples of two- or three-dimensional measurements which could not be processed by conventional means of image processing. The altimetric measurements are sparse, following the orbital path of a satellite; the ship-based measurements are irregular and highly sparse, based on the paths that a ship followed in towing an instrument array; the MRI measurements are dense, but at poor resolution and with substantial noise.

3. In many cases of scientific imaging, the raw measurement produced by an instrument is *not* a direct measurement of the multidimensional field, but rather some function of it. For example, in Application 3 we wish to study atmospheric temperature based on radiometric measurements of microwave intensities: the air temperature and microwave intensity are indeed related, but are very different quantities.

   Standard methods in image processing normally assume that the measurements (possibly noisy, possibly blurred) form an image. However, having measurements being some complicated function of the field of interest (an inverse problem) is more subtle and requires a careful formulation.

Statistics and Random Fields

What is it that makes a problem statistical, and why do we choose to focus on statistical methods?

An interest in *spatial statistics* goes considerably beyond the modelling of phenomena which are inherently *random*. In particular, multidimensional random fields offer the following advantages:

1. Even if an underlying process is not random, in most cases measurements of the process are corrupted by noise, and therefore a statistical representation may be appropriate.

2. Many processes exhibit a degree of irregularity or complexity that would be extremely difficult to model deterministically. Two examples are shown in Figure 1.4; although there are physics which govern the behaviour of both of these examples (e.g., the Navier–Stokes differential equation for water flow) the models are typically highly complex, containing a great number of unknown parameters, and are computationally difficult to simulate.

   A random-fields approach, on the other hand, would implicitly approximate these complex models on the basis of observed statistics.

A random field[1] X is nothing but a large collection of random variables arranged on some set of points (possibly a two- or three-dimensional grid, perhaps on a sphere, or perhaps irregularly distributed in a high-dimensional space). The random field is characterized by the statistical interrelationships between its random variables.

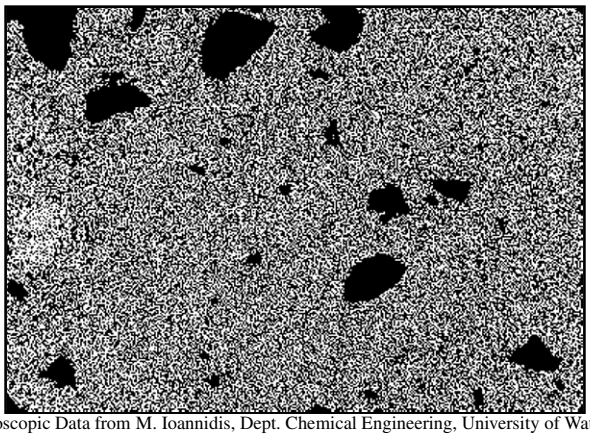
The main problem associated with a statistical formulation is the computational complexity of the resulting solution. However, as we shall see, there exists a comprehensive set of methods and algorithms for the manipulation and efficient solving of problems involving random fields. The development of this theory and of associated algorithms is the fundamental goal of this text.

Specifically, the key problem explored in this text is representational and computational efficiency in the solving of large problems. The question of efficiency is easily motivated: even a very modestly sized 256×256 image has 65 536 elements, and the glass beads image in Figure 1.4 contains in excess of 100 million elements! It comes as no surprise that a great part of the research into random fields involves the discovery or definition of *implicit* statistical forms which lead to effective or faithful representations of the true statistics, while admitting computationally efficient algorithms.

Broadly speaking there are four typical problems associated with random fields [112]:

[1] Random variables, random vectors, and random fields are reviewed in Appendix B.1.

A Porous Medium of Packed Glass Beads



(Microscopic Data from M. Ioannidis, Dept. Chemical Engineering, University of Waterloo)

Global Ocean Surface Temperature

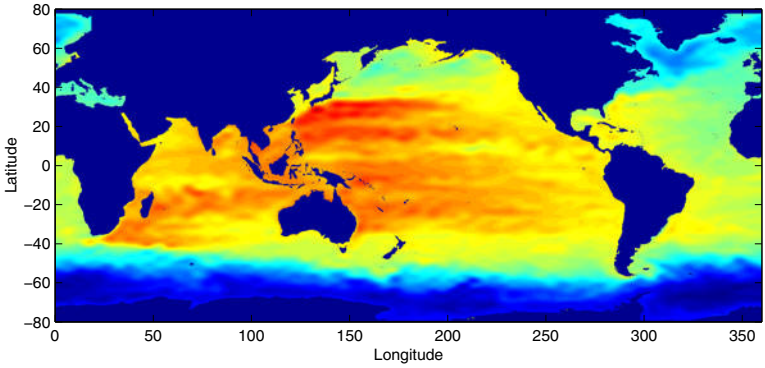


Fig. 1.4. Two examples of phenomena which may be modelled via random fields: packed glass beads (top), and the ocean surface temperature (bottom). Alternatives to random fields do exist to model these phenomena, such as ballistics methods for the glass beads, and coupled differential equations for the ocean, however such approaches would be greatly more complex than approximating the observed phenomena on the basis of inferred spatial statistics.

1. Representation: how is the random field represented and parametrized?
2. Synthesis: how can we generate "typical" realizations of the random field?
3. Parameter estimation: given a parametrized statistical model and sample image, how can we estimate the unknown parameters in the model?

4. Random fields estimation: given noisy observations of the random field, how can the unknown random field be estimated?

All four of these issues are of interest to us, and are developed throughout the text.

For each of these there are separate questions of formulation,

How do I write down the equations that need to be solved?

as opposed to those of solution,

How do I actually find a solution to these equations?

Part I of this text focuses mostly on the former question, establishing the mathematical fundamentals that are needed to express a solution, *in principle*. This gives us a solution which we might call

1. **Brute Force:** The direct implementation of the solution equations, irrespective of computational storage, complexity, and numerical robustness issues.

Parts II and III then examine the latter question, seeking practical, elegant, or indirect solutions to the problems of interest. However, *practical* should not be interpreted to mean that the material is only of dry interest to the specialist sitting at a computer, about to develop a computer program. Many of the most fundamental ideas expressed in this text are particularly in Part II, where deep insights into the nature of spatial random fields are explored.

A few kinds of efficient solutions, alternatives to the direct implementations from Part I, are summarized as follows:

2. **Dimensionality Reduction:** Transforming a problem into one or more lower-dimensional problems.
3. **Change of Basis:** A mathematical transformation of the problem which simplifies its computational or numerical complexity.
4. **Approximate Solution:** An approximation to the exact analytical solution.
5. **Approximated Problem:** Rather than solving the given problem, identifying a similar problem which can be solved exactly.
6. **Special Cases:** Circumstances in which the statistics or symmetry of the problem gives rise to special, efficient solutions.

These six points give a broad sense of what this text is about.

Interpolation as a Multidimensional Statistical Problem

We conclude the Introduction by developing a simple canonical example, to which we frequently refer throughout the text. We have chosen this problem because it is intuitive and simple to understand, yet possesses most of the features of a large, challenging, estimation problem.

Suppose you had sparse, three-dimensional measurements of some scalar quantity, such as the temperature throughout some part of an ocean. You wish to produce a dense map (really, a volume) of the temperature, based on the observed measurements.

Essentially this is an interpolation problem, in that we wish to take sparse measurements of temperature, and infer from them a dense grid of temperature values. However by *interpolation* we do not mean standard deterministic approaches such as linear, bilinear, or B-spline interpolation, in which a given set of points is deterministically projected onto a finer grid. Rather, we mean the *statistical* problem, in which we have a three-dimensional random field Z with associated measurements M, where the measurements are subject to noise V, such that

$$m_i = z_{j_i} + v_i, \tag{1.1}$$

where j_i is an index, describing the location of the ith measurement. Thus (1.1) gives the forward model, which we wish to invert (Chapter 2).

Given the definition of the inverse problem, we can formulate the analytical solution, depending on whether this is a static problem, a single snapshot in time (Chapter 3), or a more complicated time-dynamic problem, in which the temperature evolves and is estimated over time (Chapter 4).

However, so far we haven't said anything about the mathematics or statistics governing Z. What distinguishes statistical interpolation from deterministic methods, such as linear or bilinear interpolation, is the ability to take into account specific properties of Z (Chapter 5). Thus is Z smooth, on what length scales does it exhibit variability, and what happens at its boundaries? Furthermore, are the statistics of Z spatially stationary (not varying from one location to another) or not, and are the statistics best characterized by looking at correlations of Z or at the inverse correlations (Chapter 6)? Finally are there hidden underlying aspects to the problem, such that the model in one location may be different from that in another (Chapter 7)?

If the problem is particularly large, would it be possible to collapse it along one dimension, or possibly to solve the problem in pieces, rather than as a whole? One could also imagine transforming the problem, for example using a Fourier or wavelet transform (Chapter 8).

At this point we have determined what sort of problem we have, whether reduced in dimensionality, whether transformed, whether stationary. We are left with two basic

approaches for solving the inverse problem: we can convert the inverse problem to a linear system, and use one of a number of linear systems solvers (mostly iterative) to find the desired map (Chapter 9), or we could use a domain-decomposition approach that tackles the problem row-by-row, column-by-column, block-by-block, or scale-by-scale (Chapter 10). We may also wish to understand the model better by generating random samples from it (Chapter 11).

How to Read This Text

The preceding interpolation example has been very short and many details are omitted, but it is hoped that it gives the reader a sense of the scope of the ideas developed in this text. The reader wishing to follow up on interpolation in more detail is encouraged to move directly to the three interpolation examples developed in Chapter 2 on pages 20, 32, and 36.

Those readers unfamiliar with the contents of this text may wish to survey the book by glancing through the worked applications at the end of every chapter, which cover a variety of topics in remote sensing and scientific imaging. These applications, and also the various examples throughout the text, are all listed beginning on page XIII.

Any reader who wishes to explore multidimensional random fields and processes in some depth should focus on the chapters on inverse problems and modelling, Chapters 2, 5, 6, and 8, which form the core of this text.

Readers who are interested in numerical implementations of the methods in this text should consult the list of MATLAB[2] code samples on page XV. The code samples are cross-referenced to figures and examples throughout the text, and all of the listed samples are available online at

http://ocho.uwaterloo.ca/book

[2] MATLAB® is a registered trademark of The MarhWorks Inc.

Part I

Inverse Problems and Estimation

2

Inverse Problems

An understanding of forward and inverse problems [12, 301] lies at the heart of any large estimation problem.

Abstractly, most physical systems can be defined or parametrized in terms of a set of attributes, or unknowns, from which other attributes, or measurements, can be inferred. In other words, the quantities $\underline{m}$ which we measure are some mathematical function

$$\underline{m} = f(\underline{z}) \tag{2.1}$$

of other, more basic, underlying quantities $\underline{z}$, where f may be deterministic or stochastic. In the special case when f is linear, a case of considerable interest to us, then (2.1) may be expressed as

$$\underline{m} = C\underline{z} \quad \text{or} \quad \underline{m} = C\underline{z} + \underline{v} \tag{2.2}$$

for the deterministic or stochastic cases, respectively. Normally $\underline{z}$ is an ideal, complete representation of the system: detailed, noise-free, and regularly structured (e.g., pixellated), whereas the measurements $\underline{m}$ are incomplete and approximate: possibly noise-corrupted, irregularly structured, limited in number, or somehow limited by the physics of the measuring device.

The task of inferring or computing the measurements $\underline{m}$ from the detailed fundamental quantities $\underline{z}$ is known as the forward problem. This is, by definition, an *easy* problem, since the relationship between the fundamental and measured quantities is given by some model $f()$. For example knowing $\underline{z}$ — the exact shape, size, and arrangement of all blood vessels, organs, bones, etc. in the body — makes it fairly easy to predict the appearance of $\underline{m}$, a measured X-ray [183]. A variety of further examples is shown in Table 2.1.

For deterministic systems, the algorithmic task of the forward problem is referred to as "simulation;" for the stochastic counterparts the task is known as "sample-path generation" or "sampling."

P. Fieguth, *Statistical Image Processing and Multidimensional Modeling*, Information Science and Statistics, DOI 10.1007/978-1-4419-7294-1_2, © Springer Science+Business Media, LLC 2011

Knowing . . .		*It is easy to determine . . .*
The arrangement and sizes of all organs, bones, blood vessels etc.	$\longrightarrow$	The appearance of a measured X-ray, MRI, CAT scan etc.
The known masses and positions of stars in a cluster	$\longrightarrow$	The shape of the cluster's gravitational field
The 3D density of rock, magma, and minerals below the ocean floor	$\longrightarrow$	The gravitationally induced shape of the surface of the ocean
The underground distribution and layering of rock, clay, sand	$\longrightarrow$	Groundwater and other pollutant flows
A sharp image, in focus	$\longrightarrow$	A blurry image, out of focus

Table 2.1. Examples of forward problems: Generally, knowing some regular, well-structured, fundamental set of quantities (left) allows derived quantities (right) to be inferred relatively easily.

	Forward Problem: Easy	
Fundamental	$\longrightarrow$	Derived
Regularly Structured		Typically Irregular
Complete, Accurate	$\longleftarrow$	Incomplete, Approximate
	Inverse Problem: Hard	

Table 2.2. An inverse problem is, by definition, the difficult inversion of a comparatively-straightforward forward problem. Inverse problems are common because the available measurements in any given problem normally have the attributes on the right, whereas what we *want* are those are those on the left.

The task of inferring the reverse, that is, inferring the fundamental quantities $\underline{z}$ from the derived or measured ones $\underline{m}$, is known as an inverse problem. Since inferring unknowns from measurements is a universal objective in experimental science, the solution of inverse problems is of vast interest and well-discussed in the literature [24, 27, 129, 140, 154, 189, 298, 304, 308–310]. It is also a *hard* problem, for two reasons:

1. The inverse function
$$\underline{z} = f^{-1}(\underline{m}) \tag{2.3}$$
is normally not known, explicitly, from the mathematics or physics of the problem. The relationship from measurements to fundamental quantities is known only implicitly, indirectly, through the forward problem. Continuing the previous example, it takes considerable practice and skill to reconstruct $\underline{z}$, the three-dimensional anatomy of a patient, from $\underline{m}$, a sequence of observed X-ray images.

   If the forward problem is, indeed, invertible, then the inverse can be characterized as
$$\underline{z} = f^{-1}(\underline{m}) \triangleq \{\underline{z} | f(\underline{z}) = \underline{m}\}, \tag{2.4}$$
that is, to try all possible values of $\underline{z}$ to find the one which produced the observed measurements $\underline{m}$. Keep in mind that for an $n \times n$ eight-bit greyscale image, the number of possible configurations for $\underline{z}$ is enormous:
$$|\{\underline{z}\}| = \left(2^8\right)^{\left(n^2\right)}. \tag{2.5}$$
   It is important to note, however, that some inverse problems *can* be solved by the repeated application of a forward problem, such as the patch-based methods in Section 11.4, or in simple cases where a sequence $\underline{z}_i$ can be found such that $f(\underline{z}_i) \rightarrow \underline{m}$, such as in Example 2.1.

2. Normally the quantity or quality of the measurements is inadequate to allow a reconstruction of $\underline{z}$, implying that the inverse function f^{-1} does not even exist.

   For example, a single X-ray image, as sketched in Figure 2.1, cannot tell a physician whether a bone lies towards the front or the back of a body — both scenarios result in the same measurement.

Given the basic structure of an inverse problem, as outlined above, there are four questions to discuss:

1. Can we generalize this idea of forward and inverse problems to more heterogeneous problems?
2. When is the inverse problem "well-posed"? That is, does a solution exist?
3. When is the inverse problem "well-conditioned"? That is, will we find a meaningful solution, given a limited numerical accuracy in our computations?

Example 2.1: Root Finding is an Inverse Problem

Given a nonlinear function

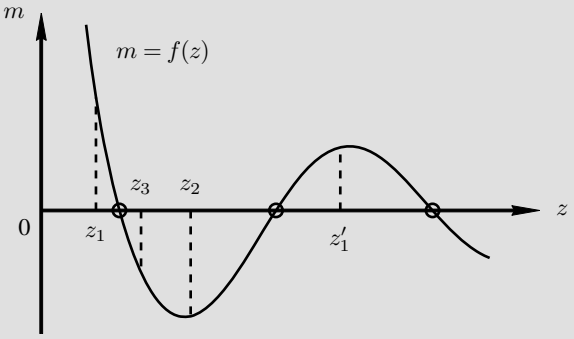
$$m = f(z),$$

finding the roots of $f()$ is just an inverse problem:

$$z = f^{-1}(0) \quad \text{or} \quad \text{Find } z \text{ such that } \; f(z) = m \equiv 0. \tag{2.6}$$

For this type of problem it is very common, in fact, for the inverse problem to be solved by repeated application of the forward problem.

For example, the bisection method of numerical root finding examines the sign of $f(z)$ for various z, starting with two points z_1, z_2 such that $f(z_1) \cdot f(z_2) < 0$:



For this inverse problem uniqueness fails — there are multiple solutions — therefore the numerical solution will be a function of initialization: starting at (z_1, z_2) will yield a different solution from starting at (z_1', z_2).

4. Given a poorly-posed or poorly-conditioned inverse problem, how do we regularize it to allow a solution to be found?

The following four sections discuss each of these, in sequence.

2.1 Data Fusion

Data fusion means many different things to different people and research disciplines. Invariably it implies some sort of fusion, or combining, of different pieces of information to infer some related quantity.

In many contexts data fusion is associated with theories of evidence and belief, especially Dempster–Shafer theory [283, 344], which seeks to separate notions of prob-

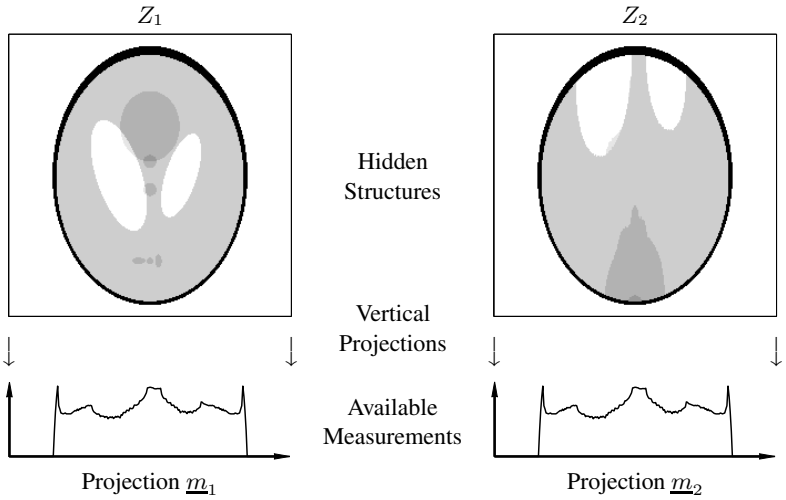


Fig. 2.1. The ambiguity of an inverse problem: One-dimensional projections (essentially line integrals) $\underline{m}_1, \underline{m}_2$ are observed for two different two-dimensional structures Z_1, Z_2. Although the two-dimensional structures are clearly very different, they result in identical projections (although clearly a sideways projection would reveal the differences!).

ability (which indicates a degree of likelihood) and belief (indicating the degree to which we are confident in the truth of some statement).

In this book we do not use Dempster–Shafer theory or notions of belief; rather, we use data fusion to mean the use of *multiple, different* measurement sets, such as shown in Figure 2.2, in solving an inverse problem. Such problems occur widely in multisensor scenarios:

- Remote sensing based on data from multiple satellite platforms
- Multispectral data processing
- Visual and infrared computer vision
- Vision, acoustic, and tactile sensors in robotics

The mathematical formulation of such a problem is no different from before: the multiple measurements

$$\underline{m}_i = f_i(\underline{z}) \tag{2.7}$$

are equivalent to a single stacked set of measurements

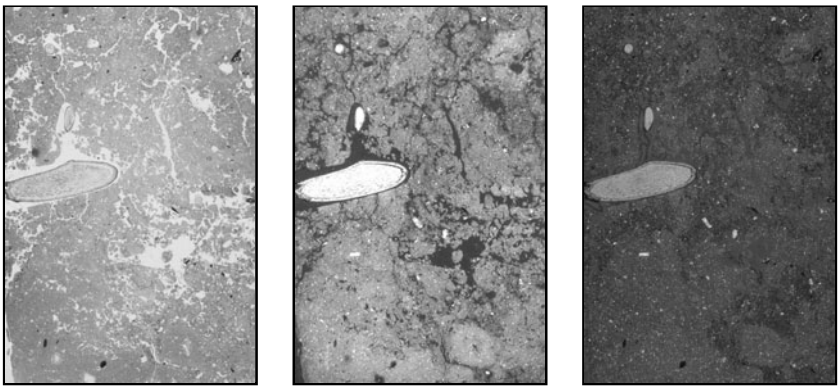


Fig. 2.2. Data fusion means the combining of multiple datasets, a problem encountered very frequently in medical imaging, remote sensing, and scientific image processing. Here a single soil sample is viewed in different polarizations, such that each image reveals different structures or attributes.

$$\underline{m} = \begin{bmatrix} \underline{m}_1 \\ \vdots \\ \underline{m}_k \end{bmatrix} = \begin{bmatrix} f_1(\underline{z}) \\ \vdots \\ f_k(\underline{z}) \end{bmatrix} = \underline{f}(\underline{z}) \tag{2.8}$$

as in (2.1).

The reason that studying data fusion is of practical interest is that the algorithmic approach to the stacked problem (2.8) may be considerably more difficult than the individual problems of (2.7). Specifically, the tractability of the individual problems of (2.7) may rely on specific properties of f_i, such as locality, sparsity, or stationarity, properties which may be lost in the combined setting.

To illustrate this point, consider the following examples from image processing:

Forward Operation		Given Measurements		Problem Solution (Inversion)
Blurring	$\Rightarrow$	Blurry Image	$\Rightarrow$	Deconvolution
Added Noise	$\Rightarrow$	Noisy Image	$\Rightarrow$	Denoising
Both	$\Rightarrow$	$\left\{ \begin{array}{c} \text{Some blurred images} \\ \text{Some noisy images} \end{array} \right\}$	$\Rightarrow$	???

That is, standard methods in image processing (Appendix C), such as deconvolution or denoising, do not extrapolate to mixed, nonstationary, heterogeneous settings. Al-

though it is not possible to formulate a single computationally efficient generalized approach to heterogeneous inverse problems, we are nevertheless motivated to find approaches to such data fusion problems.

In the following sections we do not rely on specific properties of the forward problem f; we will keep the data fusion approach in mind, and will think of f as referring to the heterogeneous case of (2.8).

2.2 Posedness

The question of posedness is one of invertibility. That is, does (2.1) admit a definition of the inverse function f^{-1}? In the early 1900s, Hadamard [154] formulated the following three criteria to be satisfied in order for a problem to be well-posed:

> EXISTENCE: Every observation $\underline{m}$ has at least one corresponding value of $\underline{z}$.
> UNIQUENESS: For every observation $\underline{m}$, the solution for $\underline{z}$ is unique.
> CONTINUITY: The dependence of the solution $\underline{z}$ on $\underline{m}$ is continuous.

We can interpret the above in the context of our linear forward problem $\underline{m} = C\underline{z}$ from (2.2). Suppose that C is $k \times n$ (the reader is pointed to Appendix A.1 if the matrix terminology used here is unfamiliar):

i. If C does *not* have full *row* rank (always true if $k > n$)
   → The rows of C are linearly dependent
   → Thus $\text{Ra}(C) \subset \mathbb{R}^k$
   → Therefore there exists $\underline{m} \in \mathbb{R}^k$ such that $\underline{m} \notin \text{Ra}(C)$
   → Existence fails.

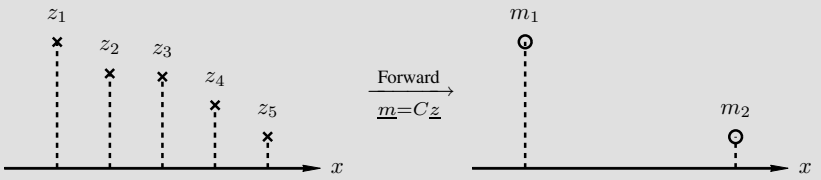
ii. If C has full *row* rank (requires $k \leq n$)
   → $\text{Ra}(C) = \mathbb{R}^k$
   → Existence is satisfied.

iii. If C does *not* have full *column* rank (always true if $k < n$)
   → The columns of C are linearly dependent
   → Thus $\text{Nu}(C) \neq \{\underline{0}\}$
   → There exists $\underline{x} \in \text{Nu}(C)$ such that $\underline{x} \neq \underline{0}$
   → It follows that $C\underline{z} = C(\underline{z} + \underline{x})$
   → Uniqueness fails.

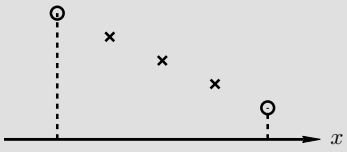
iv. If C has full *column* rank (requires $k \geq n$)
   → $\text{Nu}(C) = \{\underline{0}\}$
   → Uniqueness is satisfied.

Example 2.2: Interpolation and Posedness

Consider an inverse problem, characterized by the following forward model:



Standard linear interpolation would produce a straight-line sequence ...



But there is actually nothing inherent in the forward model to allow us to claim this kind of straight-line behaviour. Algebraically, we would write the forward model as

$$\underline{m} = C\underline{z} \quad \text{or} \quad \begin{bmatrix} m_1 \\ m_2 \end{bmatrix} = \begin{bmatrix} 1\,0\,0\,0\,0 \\ 0\,0\,0\,0\,1 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \\ z_5 \end{bmatrix} \tag{2.9}$$

Clearly $\text{Nu}(C) \neq \{\underline{0}\}$ since, for example, z_2 is not measured:

$$C \begin{bmatrix} 0 \\ z_2 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \underline{0}. \tag{2.10}$$

Therefore uniqueness fails, and the problem of interpolation is *ill-posed.*

v. If C has full row and column rank (requires $k = n$)
   → Then C is invertible
   → $\underline{z} = C^{-1}\underline{m}$ offers a single solution for $\underline{z}$ (uniqueness),
   for every value of $\underline{m}$ (existence),
   and is a continuous function of $\underline{m}$ (continuity)
   → Therefore the problem is well-posed.

The overwhelming majority of image-analysis[1] problems are ill-posed; three simple examples are illustrated in Figure 2.3:

IMAGE BLURRING: Although the blurred image may seem highly distorted from the original, if the exact nature of the blurring operation is known then it can, under certain circumstances,[2] be inverted perfectly. This problem may be well-posed.

IMAGE SUBSAMPLING: Here the problem is to interpolate the missing portions of the image. However, with pieces of the image $\underline{z}$ not observed, uniqueness fails and the problem is ill-posed.

IMAGE NOISE: Although the original image is clearly discernible through the noise, the original $\underline{z}$ is unknown and uniqueness fails.

Two further examples illustrate failed continuity and existence:

EDGE DETECTION: Suppose we wish to estimate the edge process $z(x)$ underlying an observed signal $m(x)$ [27]. We propose the inverse function

$$z(x) = \frac{\partial m}{\partial x}. \tag{2.11}$$

If we consider two sets of observations

$$m = g(z) \quad \text{and} \quad m' = g(z) + \epsilon \sin(\Omega z), \tag{2.12}$$

then m and m' can be arbitrarily close for small ϵ, however the resulting edge maps z, z' can be made arbitrarily far apart for large Ω, thus continuity fails.

REDUNDANT MEASUREMENTS: Suppose that we take multiple (redundant) measurements, for example, to reduce measurement uncertainty or noise. The nominal forward model

$$\begin{bmatrix} m_1 \\ \vdots \\ m_n \end{bmatrix} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} z \tag{2.13}$$

is ill-posed, because unless all of the measurements $m_1, \ldots, m_n$ are identical, existence fails for z.

Ill-posedness is not necessarily a terrible difficulty. Indeed, there are very simple, common problems, such as the linear regression illustrated in Example 2.3, which are ill-posed but for which solutions can be proposed. In general:

[1] Appendix C provides a review of image processing.

[2] Actually, the inversion is possible only for certain types of blur, and then only on an infinite domain, a finite domain with known boundary conditions, or on a periodic domain with a blur based on circular convolution.

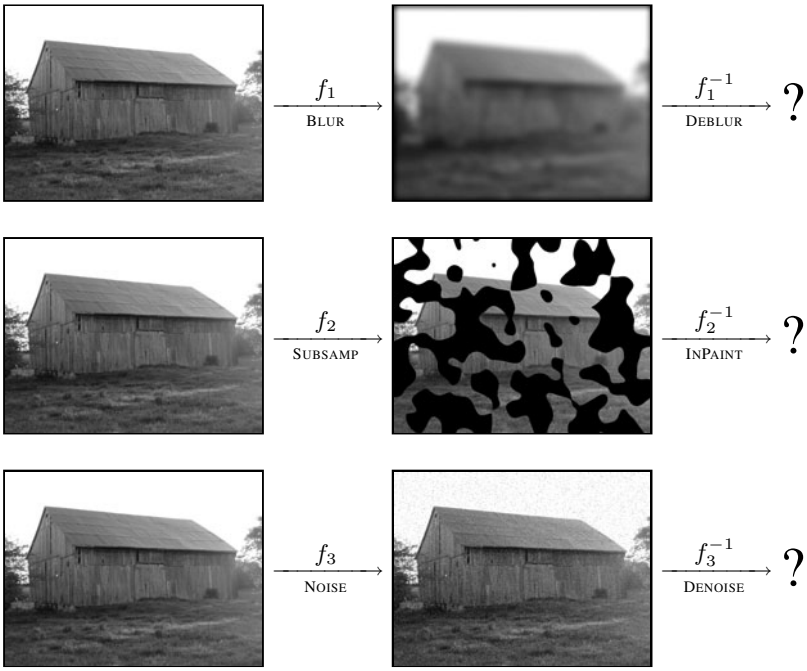


Fig. 2.3. Three simple forward operators from image processing: Blurring f_1, subsampling f_2, and additive noise f_3 (also see Appendix C). All three forward operations lead to differently-degraded images (centre column). Methods of deblurring, in-painting, and denoising are common in the image processing literature, however formally f_2 is ill-posed and f_1, f_3 may or may not be well-posed, depending on the type of image blur and noise, respectively. The question of image deblurring is further examined in the context of problem conditioning, in Figure 2.4, and the question of image denoising in the context of regularization, in Figure 2.5.

IF EXISTENCE FAILS: There is no $\underline{z}$ corresponding to the given $\underline{m}$: the problem is *overdetermined*. We view $\underline{m}$ to have been perturbed by noise from its ideal value, so we choose that $\underline{z}$ which comes closest:

Select an estimate $\hat{\underline{z}}$ by minimizing $\|\underline{m} - C\hat{\underline{z}}\|$ for some norm $\|\cdot\|$.

IF UNIQUENESS FAILS: For the given observation $\underline{m}$ there are infinitely many possible solutions for $\underline{z}$: the problem is *underdetermined*. We somehow need additional information on $\underline{z}$, such as the regularization constraints and prior models discussed later in this chapter. A simple assertion is to select a small value for $\underline{z}$:

Select an estimate $\hat{\underline{z}}$ by minimizing $\|\hat{\underline{z}}\|$ over those $\underline{z}$ satisfying $\underline{m} = C\underline{z}$.

Example 2.3: Regression and Posedness

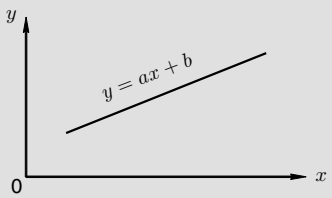
In doing linear regression, we assert a model

$$y = ax + b.$$

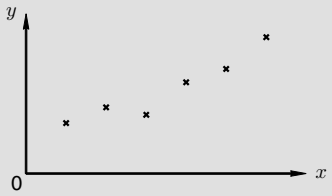
Thus we have unknowns

$$\underline{z} = \begin{bmatrix} a \\ b \end{bmatrix}.$$

In a typical linear regression we are given many $n \gg 2$ data points, but have only two unknowns a, b. This is essentially an example of repeated measurements as in (2.13), therefore *existence fails* and the problem is ill-posed.



This is easily seen in the second plot: no straight line exists which passes through all of the given points.

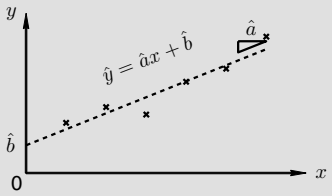


However, we never expected the given data points to lie in a perfect line; instead, we seek an estimate

$$\hat{\underline{z}} = \begin{bmatrix} \hat{a} \\ \hat{b} \end{bmatrix}$$

which passes *as closely as possible* to the given points, minimizing

$$\sum_{i=1}^{n} \Big(y_i - (\hat{a}x_i + \hat{b}) \Big)^2.$$



2.3 Conditioning

A problem $f()$ is said to be well-posed if existence, uniqueness, and continuity are satisfied. Well-posedness is a sufficient condition for the mathematical inverse $f^{-1}()$ to exist.

However, for the mathematical inverse to exist theoretically does *not* imply that it can be reliably found, in practice, using a computer with finite numerical accuracy. That is, the principal limitation in the definition of posedness is that a well-posed problem is not necessarily robust with respect to noise or numerical rounding errors.

The key question in conditioning [27,313] is the numerical precision required to produce a meaningful result, which is essentially dependent on the relative magnitudes

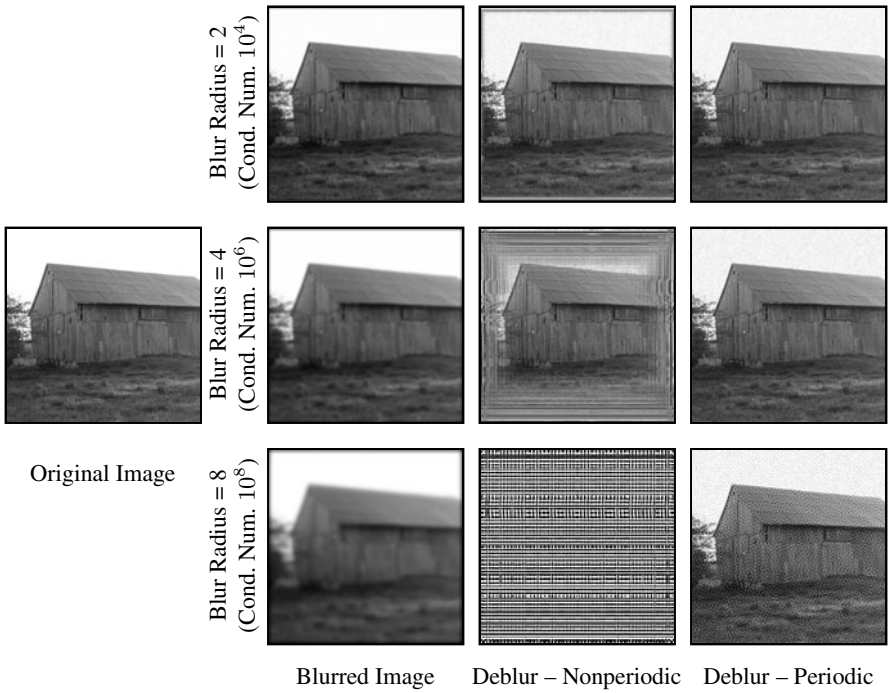


Fig. 2.4. The image deblurring problem, following up on Figure 2.3. The deblurring problem is very ill-conditioned, since blurring removes high-frequency details, and deblurring takes differences and multiplies them by a large number in order to reconstruct those details. Inasmuch as the image is not known outside of its boundaries, reconstruction errors appear first at the boundaries (middle). In the less realistic periodic case, right, both the blurring and deblurring assume the image to be periodic.

of values which need to be added or subtracted from one another. For example, the expression

$$\left[\left(10^{20} + 10^{-20}\right) - 10^{20}\right] \cdot 10^{20} \tag{2.14}$$

which obviously equals 10^0, is extremely difficult to compute numerically,[3] unless it is first simplified, yet operations such as this are very typical in working with poorly-conditioned matrices.

In the context of our canonical system $\underline{m} = C\underline{z}$, well-posedness is determined as

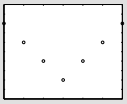
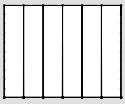
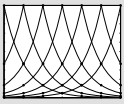
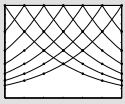
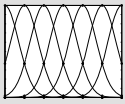
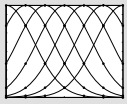
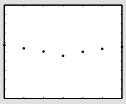
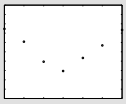
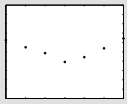
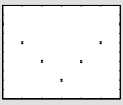
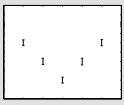
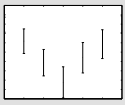
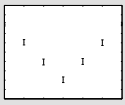
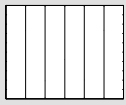
$$\text{Well Posed} \iff C \text{ is invertible} \iff \det(C) \neq 0. \tag{2.17}$$

However the *numerical* computation of $\det(C)$ is susceptible to rounding errors:

[3] MATLAB returns an answer of 0.

Example 2.4: Measurement Models and Conditioning

The degree to which measurements are blurred is a rough measure of condition number κ. In the following example five measurement structures are shown, ranging from separate measurements of each individual state element (easy) to highly overlapping, nearly redundant measurements (hard).

$\underline{z}$					
C	Delta 	Exp. 	Wide Exp. 	Gauss. 	Wide Gaus. 
$\underline{m}$					
$\kappa(C)$	1	4	23	5	745
$\hat{\underline{z}} \pm \sigma$					

Each panel in the second row shows seven curves, with each curve corresponding to one row of C. We have noisy measurements

$$\underline{m} = C\underline{z} + \underline{v} \tag{2.15}$$

which need to be inverted to form estimates:

$$\hat{\underline{z}} = C^{-1}\underline{m}. \tag{2.16}$$

Because the measurements are subject to noise, the experiment is repeated 100 times, with the results plotted as a range (the mean plus/minus one standard deviation).

The sensitivity and variability in the estimates $\hat{\underline{z}}$ to the choice of measurement model (and consequent condition number) is very clear: the larger the condition number the greater the variability (i.e., sensitivity to noise). The Gaussian blur function, in particular, is exceptionally sensitive and poorly conditioned.

Calculated $\det(C)$	Naïve Conclusion	Possible Scenario ...
0	System ill-posed	Rounding error in computation of $\det(C)$. System is actually well-posed, $\det(C) \neq 0$.
0.00043	System well-posed	Rounding error in computation of $\det(C)$. System is actually ill-posed, $\det(C) = 0$.

That is, it is actually very difficult to state, categorically, whether a matrix is singular, unless the determinant is computed analytically / algebraically.

We can quantify the degree to which such numerical errors may occur. Suppose we have a well-posed problem and perturb the observation to $\underline{m} + \delta\underline{m}$. Thus we have the modified solution

$$\underline{z} + \delta\underline{z} = C^{-1}(\underline{m} + \delta\underline{m}) \;\longrightarrow\; \delta\underline{z} = C^{-1}\delta\underline{m} \tag{2.18}$$

from which we can find bounds

$$\begin{aligned} \underline{m} = C\underline{z} \quad &\longrightarrow \quad \|\underline{m}\| \leq \|C\| \cdot \|\underline{z}\| \\ \delta\underline{z} = C^{-1}\delta\underline{m} \quad &\longrightarrow \quad \|\delta\underline{z}\| \leq \left\|C^{-1}\right\| \cdot \|\delta\underline{m}\| \,. \end{aligned} \tag{2.19}$$

We can therefore express the relative sensitivity in the estimates to perturbations in the measurements as

$$\frac{\|\delta\underline{z}\|}{\|\underline{z}\|} \leq \kappa \frac{\|\delta\underline{m}\|}{\|\underline{m}\|} \tag{2.20}$$

where, from (2.19), we find

$$\kappa \triangleq \|C\| \cdot \|C^{-1}\| \tag{2.21}$$

where κ is known as the *condition number* of matrix C.

Alternatively, suppose that the system matrix C itself is perturbed to $C + \delta C$, which induces a perturbation $\delta\underline{z}$ in the solution to the linear system. Thus

$$\underline{m} = (C + \delta C)(\underline{z} + \delta\underline{z}) = C\underline{z} + \delta C\underline{z} + C\delta\underline{z} + \delta C\delta\underline{z}. \tag{2.22}$$

Because $C\underline{z} = \underline{m}$, and ignoring the second-order term, to first order we have

$$\delta C\underline{z} = -C\delta\underline{z} \;\longrightarrow\; \delta\underline{z} = -C^{-1}\delta C\underline{z}. \tag{2.23}$$

Thus the solution sensitivity is given by

$$\frac{\|\delta\underline{z}\|}{\|\underline{z}\|} = \frac{\|-C^{-1}\delta C\underline{z}\|}{\|\underline{z}\|} \leq \frac{\|-C^{-1}\|\|\delta C\|\|\underline{z}\|}{\|\underline{z}\|} = \|-C^{-1}\|\|\delta C\|. \tag{2.24}$$

Expressing the relative sensitivity as before,

$$\frac{\|\delta\underline{z}\|}{\|\underline{z}\|} \leq \kappa \frac{\|\delta C\|}{\|C\|} \tag{2.25}$$

then

$$\kappa \frac{\|\delta C\|}{\|C\|} = \| - C^{-1}\| \|\delta C\| \quad \Rightarrow \quad \kappa \triangleq \|C^{-1}\| \|C\|, \tag{2.26}$$

which is the same notion of matrix sensitivity as before, in (2.21).

Clearly there can be various possible definitions of matrix condition number, since the expression

$$\kappa(C) = \|C\| \cdot \|C^{-1}\| \tag{2.27}$$

is a function of the chosen matrix norm, such as the large family of induced norms

$$\|C\| = \max_{\underline{x} \neq 0} \frac{\|A\underline{x}\|}{\|\underline{x}\|} \tag{2.28}$$

derived from any vector norm $\|\underline{x}\|$. If the common Frobenius norm [140] is chosen, then (2.27) reduces to the most widely used definition of condition number:

$$\kappa \triangleq \frac{\sigma_{\max}(C^{-1})}{\sigma_{\min}(C^{-1})} = \frac{\sigma_{\max}(C)}{\sigma_{\min}(C)}, \tag{2.29}$$

the ratio of largest to smallest singular values (see Appendix A.7.2) of matrix C.

Intuitively, we interpret conditioning as a generalization of posedness. For our linear system $\underline{m} = C\underline{z}$, the problem being ill-posed implies that C is rectangular or singular, further implying that $\sigma_{\min}(C) = 0$, thus $\kappa = \infty$.

In other words, we can interpret ill-posed problems as lying on one end of the spectrum of conditioning, possessing a condition number of ∞, and not materially different from a well-posed problem having a huge condition number.

Example 2.5: Matrix Conditioning

So what does the condition number κ of a matrix really mean?

Essentially, the condition number measures the degree to which a matrix is sensitive to the exact values of the matrix entries, entries which might be perturbed due to floating-point representation errors, numerical rounding errors, or numerical approximations. For example, given the two matrices

$$M_1 = \begin{bmatrix} 1 & 0 & x \\ 0 & 1 & 0 \\ x & 0 & 1 \end{bmatrix} \qquad M_2 = \begin{bmatrix} 1 & 0.99 & y \\ 0.99 & 1 & 0.99 \\ y & 0.99 & 1 \end{bmatrix} \tag{2.30}$$

Example continues ...

Example 2.5: Matrix Conditioning (cont'd)

In M_1, any value

$$-1 < x < 1 \tag{2.31}$$

is valid for x, therefore M_1 has a low condition number, since none of the matrix entries are terribly constrained by the others. On the other hand, in M_2, the value of y is forced to be in a relatively narrow range

$$0.96 < y < 1, \tag{2.32}$$

therefore the conditioning of M_2 is somewhat poor. (The reader may wish to examine (A.55) and Figure A.1 in Appendix A.4.)

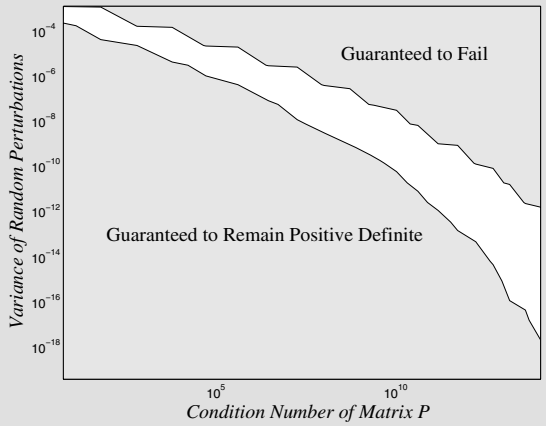
One can study condition number more systematically by simulation. Suppose we have a one-dimensional random process $\underline{z}$ with Gaussian correlation, parametrized by correlation length ξ:

$$\underline{z}(\xi) \sim P(\xi) \quad 0 < \xi. \tag{2.33}$$

Suppose we randomly distort the elements of P with a symmetric perturbation:

$$\bar{P}(\xi) = P(\xi) + (W + W^T) \quad w_{ij} \sim \mathcal{N}(0, \sigma^2). \tag{2.34}$$

By doing this operation 500 times, we can examine how often $\bar{P}$ stays positive-definite:



We see very clearly that larger condition numbers permit only smaller and smaller perturbations, if the matrix $\bar{P}$ is to stay positive-definite. That is, increasing the condition number pushes the covariance eigenvalues ever closer to zero, making it possible for ever smaller perturbations to push the eigenvalues of $\bar{P}$ across zero to be negative, in which case $\bar{P}$ fails to be positive-definite.

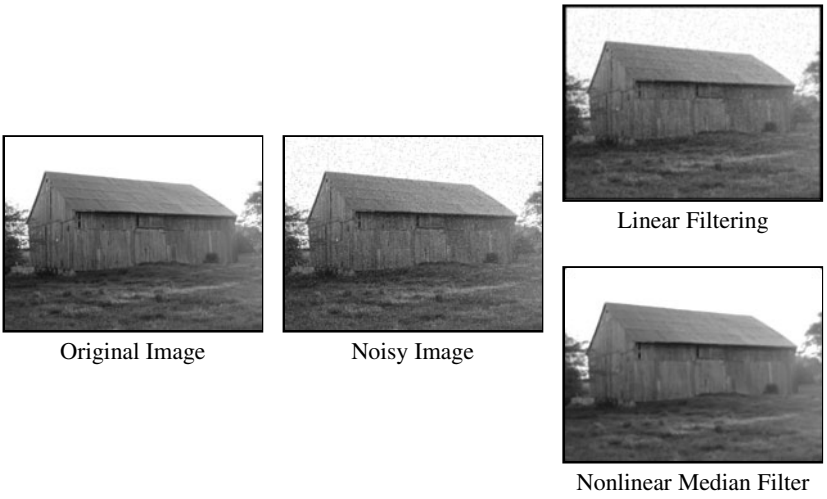


Fig. 2.5. The image denoising problem, following up on Figure 2.3. By making very modest assumptions about image smoothness, it is possible to regularize the inverse problem. A simple linear filter assumes the image to be smooth, such that the estimates are slightly blurred to attenuate the noise; a median filter, on the other hand, is a nonlinear estimator assuming the image to be piecewise planar, such that isolated noise pixels are more completely removed, and the image edges remain sharp. Further discussion can be found in Appendix C around Figure C.5.

2.4 Regularization and Prior Models

The questions of posedness and, to some extent, of conditioning are mostly theoretical ones: in our context, nearly all practical inverse problems are ill-posed. The question, then, of greatest practical interest is how to find *some* sort of meaningful solution to a given ill-posed inverse problem.

For example, although the image subsampling and additive noise examples of Figure 2.3 are ill-posed, both problems are routinely solved using interpolation, as is illustrated for the denoising problem in Figure 2.5. However the act of interpolation presupposes some degree of continuity or smoothness in $\underline{z}$, assumptions which were not explicitly stated in the model $\underline{m} = C\underline{z}$; these additional assumptions or constraints are the key to a well-posed problem.

The general approach is to take a given inverse problem and to improve the posedness or conditioning by applying additional constraints. In principle, asserting constraints to make a problem well-posed is very easy; however the key to *meaningful* regularization is to assert *just* enough constraints to adequately condition without

Ill-Posedness due to Existence Failure	
Basic Circumstance	More measurements than unknowns
Common Example	Linear Regression
Needed Constraints?	Need to make assertion regarding the *Measurement* model; *No* knowledge needed about $\underline{z}$.

Ill-Posedness due to Uniqueness Failure	
Basic Circumstance	More unknowns than measurements
Common Example	Image Interpolation
Needed Constraints?	Need to make assertion regarding the *Prior* model; Knowledge *is* required about $\underline{z}$.

Table 2.3. A Comparison of Ill-Posedness due to Uniqueness and Existence Failures

excessively modifying the original inverse problem. The approach taken depends on the nature of the ill-posedness or ill-conditioning.

First, if *existence* fails but uniqueness is satisfied, we typically select $\underline{z}$ to be most consistent with the measurements; that is, we estimate

$$\hat{\underline{z}} = \arg_{\underline{z}} \min \|\underline{m} - f(\underline{z})\|. \tag{2.35}$$

Since uniqueness is satisfied, the most consistent $\underline{z}$ is expected[4] to be unique; therefore no prior information or additional constraints are required regarding $\underline{z}$. The problem is made solvable by asserting that the measurement model is not exact, that the measurements are subject to error. If we choose a typical squared norm $\|\cdot\| = |\cdot|^2$, then (2.35) is referred to as a *least-squares* problem.

If *uniqueness* fails, there are multiple choices of $\underline{z}$ which perfectly satisfy the forward problem. Since there are multiple choices for $\underline{z}$ satisfying the measurement model perfectly, we require constraints or prior information on $\underline{z}$ itself. A very simple constraint is to select the smallest $\underline{z}$:

$$\hat{\underline{z}} = \arg_{\underline{z}} \min\{\|\underline{z}\| \mid \underline{m} = f(\underline{z})\}. \tag{2.36}$$

In practice this approach is not very useful (see Example 2.6), so Sections 2.4.1 and 5.5 develop other constraints. At a minimum, together the constraints and measurements must satisfy uniqueness.

It is important to clearly recognize the fundamental differences between problems of existence and problems of uniqueness, as outlined in Table 2.3. The natures of the problems and corresponding solutions are quite different.

[4] Whether there are one or more most consistent $\underline{z}$ will depend on the details of $f()$ and the choice of norm. The most consistent $\underline{z}$ will be unique for linear f and quadratic norm.

Questions of *continuity* or *conditioning* are a bit more subtle, addressed by *regularization*. The idea behind regularization stems from a proposal of Tikhonov [308–310], which was to design a *family* of estimators $\underline{\hat{z}}(\underline{m}, \lambda)$ which continuously approximate some ideal (ill-conditioned or discontinuous) estimator $\underline{\hat{z}}(\underline{m})$, such that

$$\lim_{\lambda \to 0} \underline{\hat{z}}(\underline{m}, \lambda) = \underline{\hat{z}}(\underline{m}). \tag{2.37}$$

In the context of this book, our specific interest is in linear least-square inverse problems, in which case the regularized estimator, combining (2.35) and (2.36), becomes

$$\begin{aligned} \underline{\hat{z}}(\underline{m}, \lambda) &= \arg_{\underline{z}} \min\big\{(\text{Meas. Constraints}) + \lambda\,(\text{Estimate Constraints})\big\} && (2.38)\\ &= \arg_{\underline{z}} \min\big\{\|\underline{m} - C\underline{z}\|_{R^{-1}} + \lambda\,\|L\underline{z}\|\big\}, && (2.39) \end{aligned}$$

where each row of L asserts a constraint, separate from the measurements, and where

$$\begin{aligned} \|\underline{m} - C\underline{z}\|_{R^{-1}} &\triangleq (\underline{m} - C\underline{z})^T R^{-1} (\underline{m} - C\underline{z}) && (2.40)\\ \|L\underline{z}\| \equiv \|L\underline{z}\|_{I^{-1}} &\triangleq (L\underline{z})^T I^{-1} (L\underline{z}) = \underline{z}^T (L^T L)\underline{z}, && (2.41) \end{aligned}$$

where R represents the measurement uncertainty.

The idea is that λ controls the degree of approximation:

Tiny λ: Weak constraints, faithful to the original, but poorly conditioned;
Large λ: Strong constraints, a poor approximation, but well conditioned.

This leaves us with two questions: how do we find appropriate constraints L, and how is an appropriate value of λ to be selected?

How we interpret and answer these questions depends on our basic understanding of the problem, leading to one of the most significant philosophical divides in statistics:

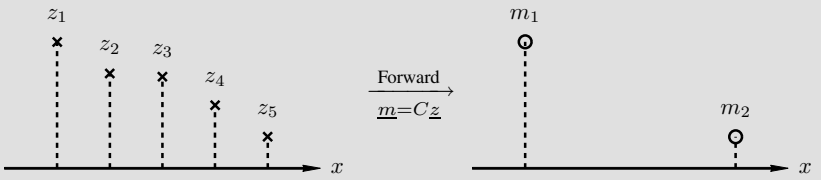
DETERMINISTIC: The vector $\underline{z}$ is just a set of *unknowns*, a set of numbers which we need to estimate.

BAYESIAN: The vector $\underline{z}$ is a set of *random variables*, which obey certain statistics given by a prior model.

In many ways this division is nothing more than philosophical, in that mathematically the two approaches can result in the same algebraic formulation, as discussed in Section 3.2.4. However, the choice of approach may influence how we set up and understand an estimation problem, outlined in the following two sections.

Example 2.6: Interpolation and Regularization

Given the inverse problem from Example 2.2,



with forward model

$$\underline{m} = C\underline{z} \quad \text{or} \quad \begin{bmatrix} m_1 \\ m_2 \end{bmatrix} = \begin{bmatrix} 1\,0\,0\,0\,0 \\ 0\,0\,0\,0\,1 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \\ z_5 \end{bmatrix} \tag{2.42}$$

there is a wide variety of approaches which we could now try to follow, in order to generate a solution:

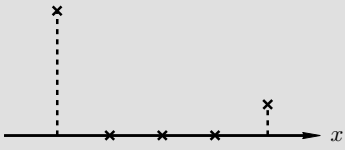
(i) We could try to solve for $\underline{\hat{z}}$ as

$$\underline{\hat{z}} = \arg_{\underline{\hat{z}}} \min \|\underline{m} - C\underline{\hat{z}}\| \quad \Rightarrow \quad \begin{array}{c}\text{Problem: Uniqueness fails} \\ \text{(as we saw in Example 2.2)}\end{array}$$

(ii) We could modify the criterion based on (2.36) and regularize:

$$\underline{\hat{z}} = \arg_{\underline{\hat{z}}} \min \left\{ \|\underline{m} - C\underline{\hat{z}}\| + \lambda \|\underline{\hat{z}}\| \right\} \quad \Rightarrow \quad \text{Well posed!}$$

The solution for $\underline{\hat{z}}$ then looks like



What went wrong? The regularization constraint penalized the deviation of individual elements of $\underline{\hat{z}}$ from zero, therefore the unmeasured points were just set to zero.

We need constraints which *inter-relate* the elements of $\underline{z}$.

Example continues . . .

Example 2.6: Interpolation and Regularization (cont'd)

(iii) To have constraints which inter-relate state elements, we can penalize the *difference* between adjacent elements, essentially a penalty on slope:

$$\underline{\hat{z}} = \arg_{\underline{\hat{z}}} \min \left\{ \|\underline{m} - C\underline{\hat{z}}\| + \lambda \|L\underline{\hat{z}}\| \right\} \tag{2.43}$$

where

$$L = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 \end{bmatrix}$$

If we select $\lambda = 1$, then the solution (2.43) is found to be

$$\underline{\hat{z}} = \begin{bmatrix} 5/6 & 1/6 \\ 4/6 & 2/6 \\ 3/6 & 3/6 \\ 2/6 & 4/6 \\ 1/6 & 5/6 \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \end{bmatrix} =$$

Now we're interpolating!

Observe, however, that $\hat{z}_1 \neq m_1, \hat{z}_2 \neq m_2$ because of the compromise in (2.43) between satisfying the measurement constraint

$$(\hat{z}_1 - m_1)^2 + (\hat{z}_2 - m_2)^2$$

and the slope constraint

$$\sum_i (\hat{z}_i - \hat{z}_{i+1})^2.$$

We can observe the effects of this tradeoff by varying λ:

Small λ	Medium λ	Large λ
Meas. Dominate	Balanced	Slope Dominates

At this point it is not, however, possible to talk about the *correct* or *optimum* value of λ. In principle, all possible tradeoffs between measurement and prior yield a valid estimate. Inferring the optimum λ on the basis on the measurements is the subject of validation, discussed in Example 2.8.

Example continues . . .

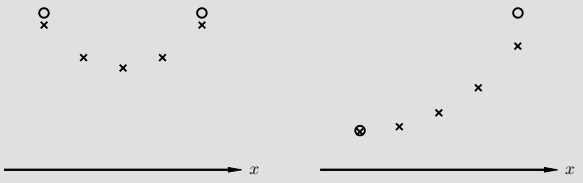
Example 2.6: Interpolation and Regularization (cont'd)

The power of this approach, in comparison to standard interpolation, is in its flexibility:

(iv) Suppose we believe that $\underline{z}$ is smooth, but should decay to zero away from measurements:

$$\underline{\hat{z}} = \arg_{\underline{\hat{z}}} \min \Big\{ \|\underline{m} - C\underline{\hat{z}}\| + \lambda\alpha \|L\underline{\hat{z}}\| + \lambda(1-\alpha) \|\underline{\hat{z}}\| \Big\}. \tag{2.44}$$

This is no longer linear interpolation:



In both cases we see intermediate points pulled towards zero.

(v) We could, in principle, have a space-varying slope penalty:

$$L = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ 0 & 2 & -2 & 0 & 0 \\ 0 & 0 & 3 & -3 & 0 \\ 0 & 0 & 0 & 4 & -4 \end{bmatrix}$$

which would lead to the following estimates:

having a high penalty (low slope) to the right, and a comparatively low penalty (permitting high slope) to the left.

2.4.1 Deterministic Regularization

The vector $\underline{z}$ is just a set of *unknowns*, a set of numbers which we need to estimate. If the unknowns are small in number and existence is satisfied, then the usual formulation of (2.35) may be adequate. However when uniqueness fails and for the large spatial problems of interest to us, some additional regularization / conditioning will

be required by specifying constraints L, leading to

$$\underline{\hat{z}} = \arg_{\underline{z}} \min\{ \|\underline{m} - C\underline{z}\|_{R^{-1}} + \lambda \|L\underline{z}\| \}, \tag{2.45}$$

where uniqueness is satisfied if and only if the combined measurements and regularization constraints

$$\begin{bmatrix} C \\ L \end{bmatrix} \tag{2.46}$$

have full column rank. By far the most common regularizer is a smoothness constraint. If we imagine, for notational convenience, that $z(x)$ is an unknown signal, a function of continuous spatial index x, then we can interpret a smoothness constraint as penalizing the derivatives of $z(x)$, integrated over space:

$$\|L\underline{z}\| \simeq \sum_i \int \gamma_i |z^{(i)}|^2 \, dx, \tag{2.47}$$

where $z^{(i)}$ is the ith order derivative. Considering a two-dimensional image $z(x, y)$, the most common definitions of smoothness are the first-order *membrane* constraint

$$\|L_1\underline{z}\| \simeq \iint \left\{ \left| \frac{\partial z}{\partial x} \right|^2 + \left| \frac{\partial z}{\partial y} \right|^2 \right\} dx \, dy, \tag{2.48}$$

which corresponds to the energy of a thin membrane or rubber sheet, and the second-order *thin-plate* constraint

$$\|L_2\underline{z}\| \simeq \iint \left\{ \left| \frac{\partial^2 z}{\partial x^2} \right|^2 + 2 \left| \frac{\partial^2 z}{\partial x \partial y} \right|^2 + \left| \frac{\partial^2 z}{\partial y^2} \right|^2 \right\} dx \, dy \tag{2.49}$$

which corresponds to the energy of a thin steel plate. Obviously the above easily generalize to other orders and to other numbers of dimensions, as we show in Section 5.5. In practice, of course, for a vector $\underline{z}$ the above derivatives are approximated as differences.

With a smoothness constraint chosen, the remaining objective is the selection of parameter λ. The key problem is that, by definition, there is no optimum or correct approach for selecting λ, because there is no objective criterion for it. Instead, we are required to select some heuristic, which has led to a large literature on parameter selection, validation, cross-validation, etc. [139, 140, 172, 173, 322, 323].

Since the focus of this text is on Bayesian regularization, the choice of λ does not greatly concern us, so we limit our attention to a few basic approaches [27].

VALIDATION: We select λ to relax the estimation problem to a certain degree of "badness." There are two basic alternatives:

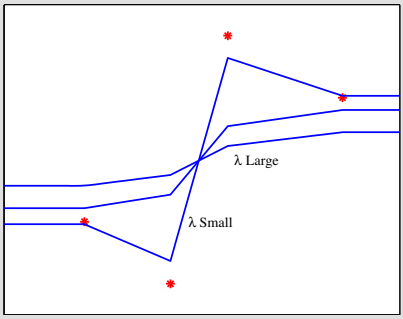
1. In (2.39), of all of the solutions $\underline{\hat{z}}$ which satisfy the constraints to some degree, select the estimate which best satisfies the *measurements*; that is,

Example 2.7: Interpolation and Smoothness Models

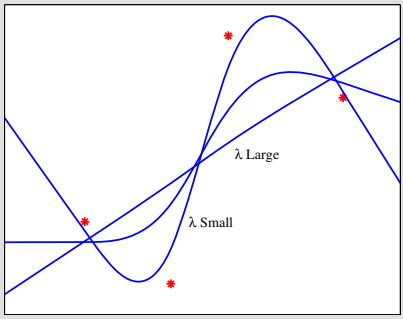
We continue with Example 2.6: we have a one-dimensional problem, with a few measurements, which we would like to interpolate.

For illustration purposes, we consider two smoothness models:

First-order example:
Penalize $\frac{\partial z}{\partial x}$



Second-order example:
Penalize $\frac{\partial^2 z}{\partial x^2}$



In each case there are four measurements, indicated by dots.

The first-order case penalizes slope (but not mean) and the solution is clearly piecewise-linear. As λ increases it tends towards a horizontal (zero-slope) line, equal to the mean of the data points.

The second-order case penalizes curvature (but not slope or mean) and is piecewise parabolic. As λ increases the solution tends towards a straight line, corresponding to the least-squares linear regression through the data points.

$$\underline{\hat{z}} = \arg_{\underline{z}} \min\left\{\|\underline{m} - C\underline{z}\|_{R^{-1}}\right\} \qquad \text{such that} \qquad \|L\underline{z}\| \leq \epsilon \tag{2.50}$$

where ϵ is a specified relaxation parameter. It can be shown [172] that this is equivalent to selecting λ such that

$$\|L\underline{\hat{z}}(\underline{m}, \lambda)\| = \epsilon. \tag{2.51}$$

2. In (2.39), of all of the solutions $\underline{\hat{z}}$ which satisfy the measurements to some degree, select the estimate which best satisfies the *constraints*; that is,

$$\underline{\hat{z}} = \arg_{\underline{z}} \min\left\{\|L\underline{z}\|\right\} \qquad \text{such that} \qquad \|\underline{m} - C\underline{z}\|_{R^{-1}} \leq \epsilon \tag{2.52}$$

where ϵ is a specified relaxation parameter. It can be shown [173] that this is equivalent to selecting λ such that

$$\|\underline{m} - C\underline{\hat{z}}(\underline{m}, \lambda)\|_{R^{-1}} = \epsilon. \tag{2.53}$$

CROSS-VALIDATION: In both of the above cases we still require the selection of parameter ϵ, which is a limited improvement over having to specify λ. In response, a variety of considerably more powerful techniques, known as cross-validation or generalized cross-validation, has been developed [322, 323], which allows the measurements to specify the most natural or fitting value of λ.

For example, suppose that we have n measurements $\underline{m}_0, \ldots, \underline{m}_{n-1}$, such that

$$\underline{m}_i = C_i \underline{z} + \underline{v}_i. \tag{2.54}$$

Then define the partial estimator $\hat{\underline{z}}_i(\lambda)$ as our usual estimator (2.39), parametrized by λ, but *omitting* data point $\underline{m}_i$:

$$\hat{\underline{z}}_i(\lambda) = \arg_{\underline{z}} \min \left\{ \sum_{j \neq i} \left\| \underline{m}_j - C_j \underline{z} \right\| + \lambda \left\| L \underline{z} \right\| \right\}. \tag{2.55}$$

Then, because $\hat{\underline{z}}_i(\lambda)$ did *not* use $\underline{m}_i$, we can meaningfully ask how close $\underline{m}_i$ is to $C_i \hat{\underline{z}}_i(\lambda)$. Ideally, if the estimator is generalizing or extrapolating sensibly from the given measurements (that is, as opposed to being corrupted by numerical errors for λ too small, or being distorted by excessive constraints for λ too large), this difference should be small, thus we can estimate λ as

$$\hat{\lambda} = \arg_{\lambda} \min \left\{ \sum_i \left\| \underline{m}_i - C_i \hat{\underline{z}}_i(\lambda) \right\| \right\}. \tag{2.56}$$

This particular form of cross-validation used a leave-one-out or "jackknife" approach, which is one of the simplest approaches from the large field of bootstrapping, bagging, and resampling theory [91, 93].

2.4.2 Bayesian Regularization

In the Bayesian setting, the vector $\underline{z}$ is a set of *random variables*. That is, the values in $\underline{z}$ are not just unknown, rather they obey certain statistics, specified by a *prior model*.

Although a wide range of prior models is possible, in the context of this book we are most interested in the second-order characterization of the unknowns: that is, specifying the prior mean $\underline{\mu}$ and covariance[5] P,

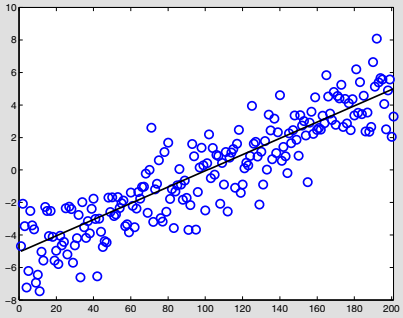
$$\underline{z} \sim (\underline{\mu}, P), \tag{2.57}$$

where, to be a valid covariance, P must be symmetric and positive-definite. If we compute the matrix square root $P = \Gamma^T \Gamma$ (see Appendix A.8), then

[5] See Appendix B.4 for a brief review of covariance matrices.

Example 2.8: Interpolation and Cross Validation

Referring back to Example 2.7, we can clearly see that the nature of the estimates is a function of the chosen penalty and associated regularization parameter. Here we use the method of cross-validation, from (2.56), to infer the best value of λ. Suppose we have a straight-line function, of which we have noisy measurements:



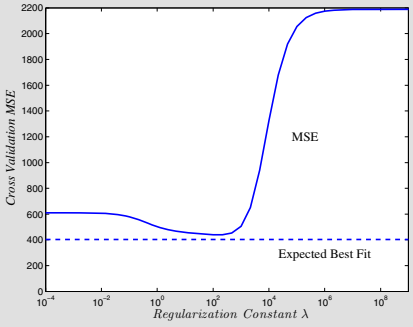
We could imagine specifying either a first-order or second-order constraint to regularize the problem:

A first-order constraint penalizes slope
$\Longrightarrow$ Our function has slope — a misfit to the constraint

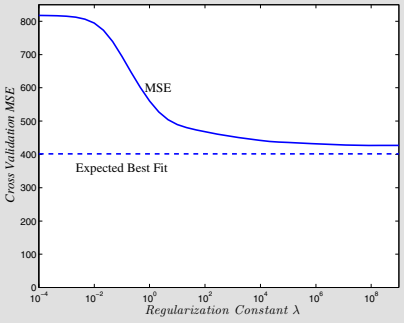
A second-order constraint penalizes curvature
$\Longrightarrow$ Our function has no curvature — perfectly consistent

Since the true function has non-zero slope, there is a tradeoff between believing the measurements and the first-order constraint; whereas the absence of curvature in the function means that there is no conflict or tradeoff between the measurements and the second-order case. We can plot (2.56) as a function of λ:

First-order constraint:



Second-order constraint:



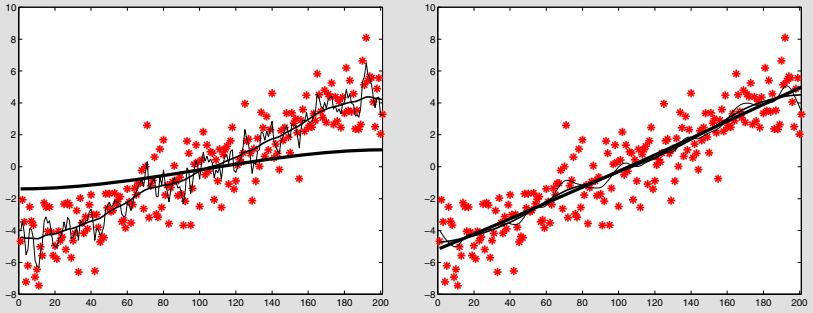
Example continues . . .

Example 2.8: Interpolation and Cross Validation (cont'd)

In the first-order case we can clearly see a unique optimum $\lambda \approx 100$, representing the best tradeoff between measurements and constraints. In contrast, the second-order case seeks an arbitrarily large λ: the more strongly the constraints are asserted, the better are the results. Below are plotted the estimates for three values of λ:

First-order
$(\lambda = 10^0, 10^2, 10^4)$

Second-order
$(\lambda = 10^2, 10^4, 10^6)$



In the first-order case, the transition from measurement overfit (thin line, λ too small), to well fit, to overconstrained (thick line, λ too large) is clearly visible. In the second-order case, the larger values of λ progressively lead to a better fit.

$$\underline{z} \sim P \longrightarrow \underline{z} \sim \Gamma^T\Gamma \longrightarrow \Gamma^{-T}\underline{z} \sim I \longrightarrow \Gamma^{-T}(\underline{z} - \underline{\mu}) = \underline{w}, \tag{2.58}$$

where $\underline{w}$ is a unit-variance, white-noise vector. That is, the covariance P implicitly specifies a set of independent constraints Γ^{-T} on $\underline{z}$; similarly, a set of constraints L implicitly specifies a sort of prior model[6] $P = (L^TL)^{-1}$.

In other words, there is a certain *duality*, or equivalence, between specifying a statistical prior covariance P for $\underline{z}$ and a set of constraints $L = \Gamma^{-T}$ on $\underline{z}$. The differences, then, between a Bayesian prior model and a set of constraints are as follows:

1. In the Bayesian case, the statistics are an inherent part of the problem, they are not just a constraint to condition the numerics. Of course, the degree to which we believe the statistics may live on a continuum:

   - High belief: the statistics are known to be correct, and can be proved from physics or derived mathematically;

[6] In many cases (L^TL) will be singular, however this does not necessarily prevent us from computing estimates.

- Medium belief: the problem obeys a prior, however the prior is complicated or unknown, so we approximate it, possibly learned from measurements;
- Low belief: the problem doesn't really obey a prior; the statistics are asserted essentially for regularization purposes.

2. In the Bayesian case, there is a unique solution to the constraint parameter λ. In order to minimize the mean-squared error

$$E\left[(\underline{z}-\underline{\hat{z}})(\underline{z}-\underline{\hat{z}})^T\right] \tag{2.59}$$

we will see (Chapter 3) that $\lambda = 1$. That is, the tradeoff between measurements and prior is not under user control, but is inherent in the relative uncertainties in the measurements and in the prior model.

2.5 Statistical Operations

The discussion thus far has been somewhat abstract, focusing on establishing an understanding of inverse problems and related issues of posedness, conditioning, regularization, and prior models.

The remainder of this chapter seeks to make the discussion considerably more specific, focusing on the three canonical problems which are of greatest interest to us, and a discussion of the types of statistical questions which we may wish to answer.

2.5.1 Canonical Problems

We are interested in large static and time-varying linear stochastic systems. The reader should observe how these problems are interconnected: the static problem is a special case of data fusion, and the static and data fusion problems are both special cases of the dynamic one.

I. THE STATIC PROBLEM: Given a single random field $\underline{z}$ obeying some prior model,

$$\underline{z} \sim \mathcal{N}(\underline{0}, P), \tag{2.60}$$

we are given a single set of measurements

$$\underline{m} = C\underline{z} + \underline{v} \qquad \underline{v} \sim \mathcal{N}(\underline{0}, R). \tag{2.61}$$

Examples of this category of problem include image enhancement, image restoration, and many applications of medical imaging.

The reader may wonder whether the above formulation is sufficiently general, since $\underline{z}$ and $\underline{v}$ may not be guaranteed to be zero mean. In fact, because our problem is linear, superposition applies, in which case the deterministic (mean) and random parts of the problem are fully decoupled. In other words, given

$$\bar{\underline{z}} \sim \mathcal{N}(\underline{\mu}, P), \tag{2.62}$$

we can formulate the mean-removed field

$$\underline{z} = \bar{\underline{z}} - \underline{\mu} \sim \mathcal{N}(\underline{0}, P). \tag{2.63}$$

After computing the estimates $\hat{\underline{z}}$ for the mean-removed field, it follows that

$$\hat{\bar{\underline{z}}} = \hat{\underline{z}} + \underline{\mu}. \tag{2.64}$$

Similarly, because any deterministic mean portion of the measurements is known ahead of time, the mean does not affect the estimates. That is, given

$$\bar{\underline{m}} = C\underline{z} + \underline{\alpha} + \underline{v} \qquad \underline{v} \sim \mathcal{N}(\underline{\beta}, R), \tag{2.65}$$

for constants $\underline{\alpha}, \underline{\beta}$ it follows that

$$\hat{\underline{z}}(\bar{\underline{m}}) = \hat{\underline{z}}(\bar{\underline{m}} - \underline{\alpha} - \underline{\beta}). \tag{2.66}$$

Thus, without loss of generality, we normally assume all mean terms to equal zero. From time to time, if it assists in clarity or intuition, the mean term may be included.

II. THE DATA FUSION PROBLEM: We are given one or more random fields $\underline{z}_i$ obeying some prior model,

$$\underline{z} = \begin{bmatrix} \underline{z}_1 \\ \vdots \\ \underline{z}_f \end{bmatrix} \sim \mathcal{N}(\underline{0}, P), \tag{2.67}$$

where the prior model may couple the fields (dense P) or leave them uncoupled (block-diagonal P).

We are given two or more sets of measurements

$$\underline{m}_i = C_i\underline{z} + \underline{v}_i \qquad \underline{v}_i \sim \mathcal{N}(\underline{0}, R_i). \tag{2.68}$$

The relationship C_i between measurements and unknowns may be highly problem-dependent.

Such data fusion problems are particularly common in remote sensing, where measurements may be available from multiple instruments on a single satellite platform, or where data from multiple satellites are to be used in studying a particular area.

III. THE DYNAMIC PROBLEM: We are given an initial prior model

$$\underline{z}(0) \sim \mathcal{N}(\underline{0}, P_0) \tag{2.69}$$

and a time-recursive dynamic model

$$\underline{z}(t+1) = A(t)\underline{z}(t) + B(t)\underline{w}(t) \qquad \underline{w}(t) \sim \mathcal{N}(\underline{0}, I) \tag{2.70}$$

where the *process noise* or driving term $\underline{w}(t)$ is white and uncorrelated with $\underline{z}$:

$$E\big[\underline{w}(t)\underline{w}(s)^T\big] = \delta_{s,t} I \qquad E\big[\underline{w}(t)\underline{z}(s)^T\big] = 0 \text{ if } t \geq s. \tag{2.71}$$

Measurements of the process arrive over time:

$$\underline{m}(t) = C(t)\underline{z}(t) + \underline{v}(t) \qquad \underline{v}(t) \sim \mathcal{N}\big(\underline{0}, R(t)\big). \tag{2.72}$$

The most obvious examples of multidimensional dynamic problems are in video data processing.

2.5.2 Prior Sampling

Given a random variable z obeying some prior probability density function (PDF) $p(z)$, sampling from the prior distribution means generating independent random samples $z_1, \ldots, z_q$ from $p(z)$. For a set $\{z_i\}$ to represent independent random samples of a distribution, it implies that any statistical question posed of random variable z and of the samples $\{z_i\}$ converge to the same value as the size of the sample set grows. In other words

$$\lim_{q \to \infty} \frac{1}{q} \sum_i f(z_i) \longrightarrow E\big[f(z)\big] \tag{2.73}$$

for any function f.

In the context of the canonical static problem (2.60), given a random vector $\underline{z}$ obeying a prior model

$$\underline{z} \sim (\underline{\mu}, P), \tag{2.74}$$

we may be interested in generating independent random samples $\underline{z}_1, \ldots, \underline{z}_q$ such that

$$\frac{1}{q} \sum_i^q \underline{z}_i \longrightarrow \underline{\mu} \qquad \frac{1}{q} \sum_i^q (\underline{z}_i - \underline{\mu})(\underline{z}_i - \underline{\mu})^T \longrightarrow P. \tag{2.75}$$

Such samples may be generated by finding Γ, the matrix square root (see Appendix A.8) of P:

$$\text{Given } \underline{w} \sim I \Rightarrow \text{cov}(\Gamma^T \underline{w}) = E\big[\Gamma^T \underline{w}\, \underline{w}^T \Gamma\big] = \Gamma^T I \Gamma = P. \tag{2.76}$$

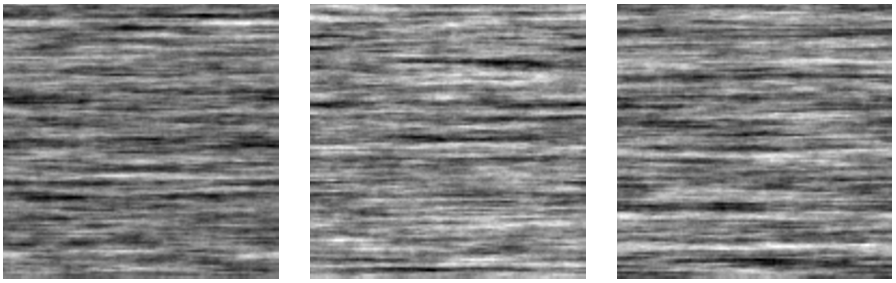


Fig. 2.6. Three samples from the prior model corresponding to a "Tree-Bark" random field. Being random samples, all three images are different, yet obey the same statistics.

Thus $\underline{\mu} + \Gamma^T \underline{w}$ is a random prior sample from model $(\underline{\mu}, P)$. Three samples from a single random field model are shown in Figure 2.6.

In the context of the canonical dynamic problem (2.70), the process of prior sampling is normally known as simulation, in that we are simulating the evolution of the dynamic process over time. The process initialization

$$\underline{z}(0) \sim \mathcal{N}(\underline{0}, P_0) \tag{2.77}$$

requires sampling $\underline{z}(0)$ from the prior model P_0, as discussed above. With this step completed, the remainder of the simulation is the straightforward recursion

$$\underline{z}(t+1) = A(t)\underline{z}(t) + B(t)\underline{w}(t). \tag{2.78}$$

Because $\underline{w}(t)$ obeys a simple prior $\mathcal{N}(\underline{0}, I)$, generating a random sample of $\underline{w}(t)$ just amounts to generating a vector of zero-mean, unit-variance Gaussian random variables.

The generation of samples from a prior model is of interest for two main reasons:

1. The samples $\underline{z}$ or $\underline{z}(t)$ depend purely on the prior model, with no dependence on any measurements. Thus the prior samples reflect the assumptions implicit in the prior model, possibly leading to insights regarding model strengths and weaknesses, based on whether the observed sample behaviour is as expected or runs counter to the desired physics or mathematics.

2. The samples $\underline{z}$ or $\underline{z}(t)$ may be of interest in and of themselves. Two common examples include image rendering (e.g., sampling of random textures from a prior texture model) and further analysis (e.g., sampling of random three-dimensional porous media for further porosity or groundwater-flow analysis).

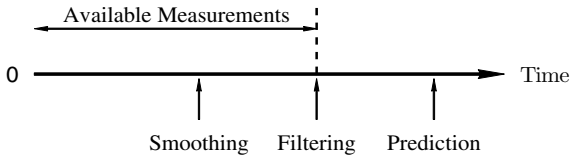


Fig. 2.7. Dynamic estimation can be broken into three inter-related problems: smoothing, filtering, and prediction, depending on whether the value to be estimated lies within, at the end, or after the end of the available measurements, respectively.

2.5.3 Estimation

Estimation is the solving of an inverse problem, the production of estimates of the state $\underline{z}$ underlying a system based on the measurements $\underline{m}$.

In the case of the *Static Problem* we seek estimates $\hat{\underline{z}}$ which are simultaneously consistent, according to some criterion, with the prior model (2.60) and with the measurement model (2.61).

In the analogous case of the *Dynamic Problem* we seek estimates $\hat{\underline{z}}(t)$ which are simultaneously consistent with the initial prior P_0, the dynamics (2.70), and the measurements (2.72). Dynamic estimation is frequently divided into the three related problems, illustrated in Figure 2.7:

1. Filtering: estimate $\hat{\underline{z}}(t)$ based on measurements $\underline{m}(s),\ 0 \leq s \leq t$.
2. Smoothing: estimate $\hat{\underline{z}}(t)$ based on measurements $\underline{m}(s),\ 0 \leq s \leq t + \tau$.
3. Prediction: estimate $\hat{\underline{z}}(t)$ based on measurements $\underline{m}(s),\ 0 \leq s \leq t - \tau$.

That is, the problem depends on the distribution of available measurements relative to the quantity being estimated. The solution to the above three is very similar, so we normally do not concern ourselves with the distinction between them and generally just refer to *estimation*.

A large number of different estimators have been proposed, for which a brief summary follows. A more detailed derivation and development of those estimators used throughout this book may be found in Chapter 3 and in [248, 284].

Bayesian Estimators

The unknown $\underline{z}$ is random for which prior information (possibly mean and covariance, higher-order statistics, or the full PDF) is available.

BAYESIAN ESTIMATOR (BE): the most general estimator,

$$\underline{\hat{z}}_B \triangleq \arg_{\underline{\hat{z}}} \min\Big\{E\big[C(\underline{z},\underline{\hat{z}})|\underline{m}\big]\Big\}, \tag{2.79}$$

finds estimates to minimize a specified cost function $C()$.

BAYESIAN LEAST-SQUARES ESTIMATOR (BLSE): a widely-used special case of the above, and normally much more tractable, is to choose a quadratic form for $C()$:

$$\underline{\hat{z}}_{\text{BLSE}} \triangleq \arg_{\underline{\hat{z}}} \min\Big\{E\big[(\underline{z}-\underline{\hat{z}})^T(\underline{z}-\underline{\hat{z}})|\underline{m}\big]\Big\}. \tag{2.80}$$

LINEAR LEAST-SQUARES ESTIMATOR (LLSE): also known as the Best Linear Unbiased Estimator (BLUE), uses the same quadratic criterion as for the BLSE, except that the estimator is required to be a linear function

$$\underline{\hat{z}}_{\text{LLSE}} = A\underline{m} + \underline{b} \tag{2.81}$$

of the measurements. That is, the effective LLSE criterion is

$$\underline{\hat{z}}_{\text{LLSE}} = A\underline{m} + \underline{b} \qquad [A,\underline{b}] = \arg_{[A,\underline{b}]} \min\Big\{E\big[(\underline{z}-A\underline{m}-\underline{b})^T(\underline{z}-A\underline{m}-\underline{b})\big]\Big\}. \tag{2.82}$$

MAXIMUM A POSTERIORI (MAP): the estimate is chosen to maximize the posterior probability density

$$\underline{\hat{z}}_{MAP} \triangleq \arg_{\underline{z}} \max p(\underline{z}|\underline{m}) = \arg_{\underline{z}} \max p(\underline{m}|\underline{z})p(\underline{z}), \tag{2.83}$$

where the latter form, derived via Bayes' rule ((B.21)), is normally the more convenient, because it expresses the estimator in terms of known quantities: the prior model $p(\underline{z})$ and the measurement model $p(\underline{m}|\underline{z})$.

KRIGING: a long-established technique of spatial estimation [75, 227, 291], the method is similar to the LLSE and BLUE, in that an estimator is sought, linear in the measurements

$$\underline{\hat{z}} = A\underline{m} + \underline{b} \tag{2.84}$$

such that

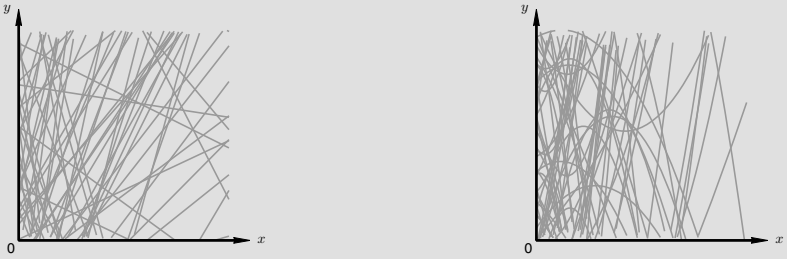
$$E[\underline{\hat{z}} - \underline{z}] = \underline{0} \qquad \text{var}(\underline{\hat{z}} - \underline{z}) \text{ is minimized,} \tag{2.85}$$

where the prior spatial model for $\underline{z}$ is specified through its *variogram*

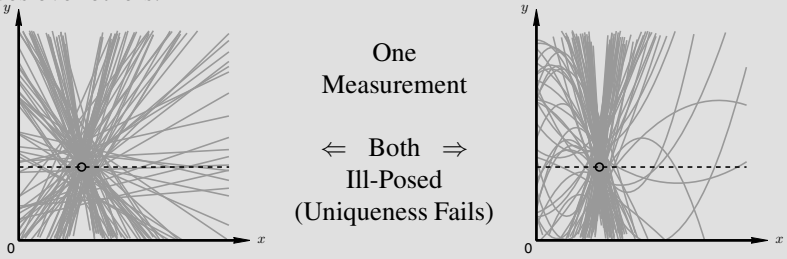
$$\gamma(\delta) = \tfrac{1}{2}\text{var}(z_1 - z_2), \tag{2.86}$$

Example 2.9: State Estimation and Sampling

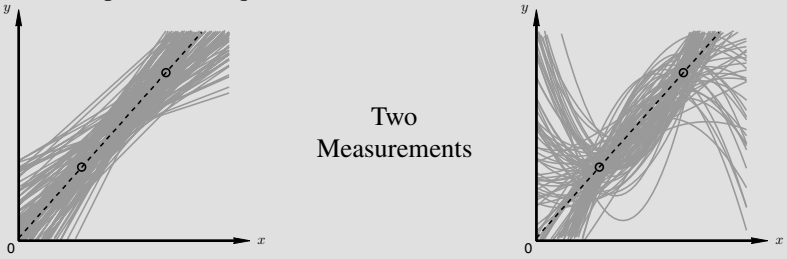
Building on Example 2.3, let's consider the infinity of all possible lines (a first-order problem, left) or parabolae (second-order, right):



The above are essentially prior samples: random samples unconstrained by measurements. A measurement is just a constraint, asserting a preference for certain curves over others:



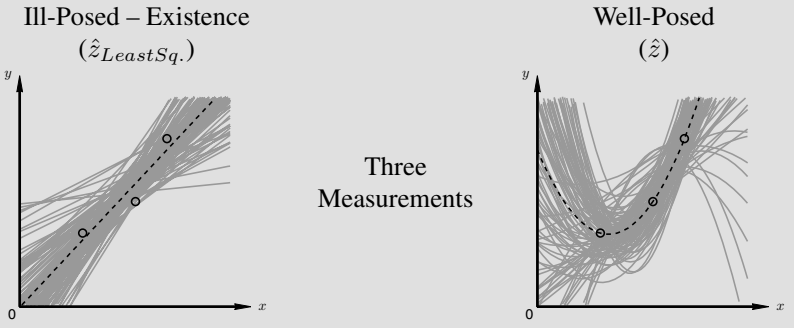
A single measurement is unable to constrain a line or a parabola, so uniqueness fails, and the dashed line plots the minimum-norm solution of (2.36). With *two* measurements the first-order problem is well-posed, however because the measurements are not exact the posterior samples still show some variability. For the second-order problem uniqueness still fails:



At three measurements the first-order problem is overconstrained (existence fails), still allowing a least-squares estimate to be computed from (2.35), whereas the second-order problem is now well-posed:

Example continues . . .

Example 2.9: State Estimation and Sampling (cont'd)



We clearly see measurements as constraining the statistics of the problem, such that more measurements increasingly concentrate the posterior samples near the estimates.

The elegance of the Bayesian formulation is that the preceding distinctions between minimum-norm or least-squares solutions are unimportant, and the estimates $\hat{z}$ (dashed curve in each panel) are calculated the same way in each case.

To better understand the calculations in this example the reader may wish to follow up with Example 3.1. Two parallel examples, showing ensembles of possible curves as a way of visualizing statistics, are shown in Example 3.4 on page 75 for the static problem, and Example 4.1 on page 92 for the dynamic problems.

where z_1 and z_2 are separated, spatially, by an amount δ.

It is significant to note that the kriging variogram encodes only a knowledge of the statistics of *differences*. That is, there is absolutely no notion of prior mean or variance.

It is significant to note that for Gaussian problems (that is, those in which $\underline{z}$ and $\underline{m}$ are jointly Gaussian), the MAP and BLSE estimators are linear, and in fact are identical to the LLSE.

Since the LLSE is the only estimator requiring no information other than the second-order statistics of the estimation problem, it is a common choice if the known statistics are limited. Even if the statistics are known in detail, and are non-Gaussian, the LLSE may still be chosen, at least as a first step, given the alternative of highly complex, normally nonlinear estimators BE, BLSE, and MAP.

Non-Bayesian Estimators

The unknown $\underline{z}$ is not random, and no statistics of $\underline{z}$ are used.

LEAST SQUARES (LS): the measurements are modelled to be a noisy function of the unknowns

$$\underline{m} = f(\underline{z}) + \underline{v}, \tag{2.87}$$

thus the unknowns are chosen to minimize the squared error in the measurements:

$$\hat{\underline{z}}_{LS} \triangleq \arg_{\underline{z}} \min \big(\underline{m} - f(\underline{z})\big)^T \big(\underline{m} - f(\underline{z})\big). \tag{2.88}$$

LINEAR LEAST SQUARES (LLS): given the linear measurements

$$\underline{m} = C\underline{z} + \underline{v}, \tag{2.89}$$

the unknowns are chosen to minimize the squared error

$$\hat{\underline{z}}_{LLS} \triangleq \arg_{\underline{z}} \min (\underline{m} - C\underline{z})^T (\underline{m} - C\underline{z}), \tag{2.90}$$

leading to an estimator $\hat{\underline{z}}_{LLS}$ a linear function of $\underline{m}$.

WEIGHTED LINEAR LEAST SQUARES (WLS): as before, however the measurement errors are not all treated equally, instead, a weighting matrix W controls the relative degree to which measurement errors are tolerated:

$$\hat{\underline{z}}_{WLS} \triangleq \arg_{\underline{z}} \min (\underline{m} - C\underline{z})^T W (\underline{m} - C\underline{z}). \tag{2.91}$$

TOTAL LEAST SQUARES (TLS): the error is modelled both in the measurements $\underline{m}$, but also in the measurement model C [141]. That is, the measurements are modelled as

$$\underline{m} = (C + E)\underline{z} + \underline{v}, \tag{2.92}$$

where E is the unknown perturbation in the measurement model. Total least squares then seeks to minimize the norm of the concatenated error $\| \begin{bmatrix} \underline{v} & E \end{bmatrix} \|_2$, which is found as the singular value decomposition of matrix $\begin{bmatrix} \underline{m} & -C \end{bmatrix}$.

MAXIMUM LIKELIHOOD (ML): the non-Bayesian analogue of MAP, the estimates are chosen to make the observed measurements most probable:

$$\hat{\underline{z}}_{\text{ML}} \triangleq \arg_{\underline{z}} \max p(\underline{m}|\underline{z}). \tag{2.93}$$

Although we are primarily concerned with Bayesian methods of estimation, we shall see (Section 3.2.4) that there is a certain duality between the Bayesian and non-Bayesian methods. As with the Bayesian methods, in the case where the measurement model is linear and the measurement noise statistics are Gaussian, then the ML estimator is linear and equivalent to the LLS, WLS estimators (for certain choices of weighting matrix and measurement error statistics).

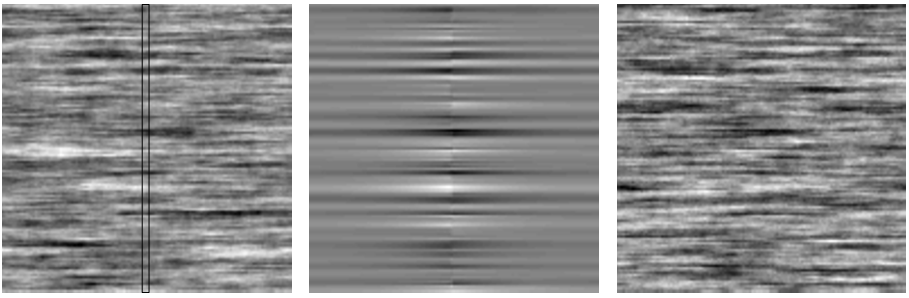


Fig. 2.8. An illustration of posterior sampling: only a sparse subset (middle columns) of a texture are measured (left), leading to estimates (middle) which lack the statistical variability of the original texture. A posterior sample (right) is consistent with the measurements, but where not constrained by the measurements follows the statistical pattern of the prior model.

2.5.4 Posterior Sampling

In Section 2.5.2 we discussed the sampling of a random vector

$$\underline{z} \sim p(\underline{z}) \tag{2.94}$$

such that the independent random samples $\underline{z}_1, \ldots, \underline{z}_q$ represent "typical" samples of the prior, without any measurement constraints.

For estimation, we seek the unique, optimum estimate $\hat{\underline{z}}$ which balances the constraints of a prior model $p(\underline{z})$ and measurements $\underline{m}$, for example

$$\hat{\underline{z}}_{\text{MAP}} = \arg_{\underline{z}} \max p(\underline{m}|\underline{z})p(\underline{z}). \tag{2.95}$$

It is important to realize that the estimate $\hat{\underline{z}}$ does *not* necessarily look like a typical or natural sample of the random field, especially if the measurements are sparse. That is, although the estimate is *optimum* under some criterion, it may *not* represent a typical or representative sample of the system being studied. This is precisely because the most likely (MAP) sample is not typical: for a Gaussian random variable z of mean μ, the most likely value of z is μ, however we do not expect a typical sample of z to precisely equal μ!

Instead, to find a typical random field, subject to the constraints of both the prior and measurements, requires that we draw a random sample from the posterior distribution, a much more subtle and difficult problem than estimation. That is, we want to draw a sample from the conditional distribution

$$(\underline{z}|\underline{m}) \sim p(\underline{z}|\underline{m}), \tag{2.96}$$

as is illustrated in Figure 2.8 and Example 2.9.

As in the case of prior sampling, samples drawn from the posterior may be desired for purposes of visualization, further analysis, or Monte Carlo studies.

2.5.5 Parameter Estimation

In some contexts the prior model may not be completely specified. Its form (e.g., Gaussian) may be known, but certain details (such as the variances or correlations between random variables, etc.) are not, and need to be estimated. That is, we have vectors of unknowns $\underline{\theta}_P, \underline{\theta}_M$ in the prior and measurement models, respectively:

$$\underline{z} \sim p(\underline{z}|\underline{\theta}_P) \qquad \underline{m} \sim p(\underline{m}|\underline{z}, \underline{\theta}_M). \tag{2.97}$$

In virtually all cases, the parameters are treated as unknowns (rather than random), and so a non-Bayesian approach, typically Maximum Likelihood, is used. That is, we select those model parameters which are most consistent with (maximize the likelihood of) the measurements:

$$\widehat{\underline{\theta}_P, \underline{\theta}_M} = \arg_{\underline{\theta}_P, \underline{\theta}_M} \max p(\underline{m}|\underline{\theta}_P, \underline{\theta}_M). \tag{2.98}$$

A full treatment of model parameter estimation is really the domain of *system identification* [212], which is well beyond the scope of this book. We are, however, interested in model identification in certain contexts where we propose a specific spatial model, such as a Markov random fields in Chapter 6 or the marching and multiscale models in Chapter 10.

Application 2: Ocean Acoustic Tomography

There are many circumstances in which we measure integrals through absorbing media, such as in medical imaging (as was illustrated in Figure 2.1 for X-rays); the process of inverting such measurements is known as *tomography*.

One recent and very novel tomographic problem is the acoustic "imaging" of the ocean [238]. Amazingly, low-frequency sound waves can propagate for *thousands* of kilometers. The key reason for the effective sound propagation is that the ocean acts as an acoustic waveguide: as shown in Figure 2.9, at a depth of approximately 1 km the sound speed is at a minimum, meaning that propagating wavefronts will bend towards 1 km depth, rather than hitting the surface or being absorbed at the ocean bottom.

Figure 2.9 sketches three different acoustic waves, all leaving from the same source and being observed at a common receiver. Observe what is happening: the different

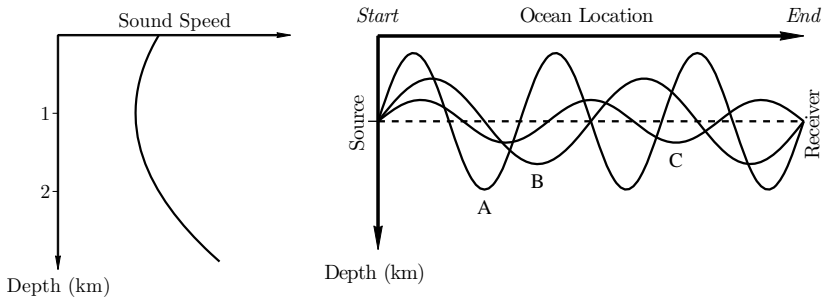


Fig. 2.9. Because the speed of sound in water has a minimum at an intermediate depth, left, the ocean acts as an acoustic waveguide, right.

waves "observe" different parts of the ocean, such that the high-amplitude wave measures a wide range of depths, whereas the low-amplitude wave is mostly sensitive to the ocean at a depth of 1 km.

What can these different sound waves tell us? Each sound wave has a different arrival time, where the arrival time is a function of the average (or integrated) sound speed over the path of the wave. The sound speed is primarily a function of depth (thus A arrives first, then B, then C), but also salinity, temperature, and ocean currents. The effect of salinity is slight, and the effect of depth can be accounted for via a sufficiently accurate ray-tracing along the lines of Figure 2.9, leaving the effects of temperature and currents to be inferred.

We now have two different inverse problems to solve, as illustrated in Figure 2.10:

1. **Acquire the tomographic line-integral measurements:**
   The measured arrival times depend on the average sound speed along the highly complex wavefront trajectories. To convert into a simpler, standard tomographic problem involving straight-line integrals, we wish to divide the ocean into layers, as shown in Figure 2.10, and to solve an inverse problem to find the average sound speed over each layer.

   That is, if ray j has an observed average sound speed of m_j, and by simulating the ray paths we know that ray j spends a fraction $a_{j,i}$ in depth layer i, then we wish to solve

$$\begin{bmatrix} m_1 \\ m_2 \\ \vdots \end{bmatrix} = A \begin{bmatrix} z_1 \\ z_2 \\ \vdots \end{bmatrix} \tag{2.99}$$

   In practice, we would probably have a number of acoustic receivers spaced at different depths.

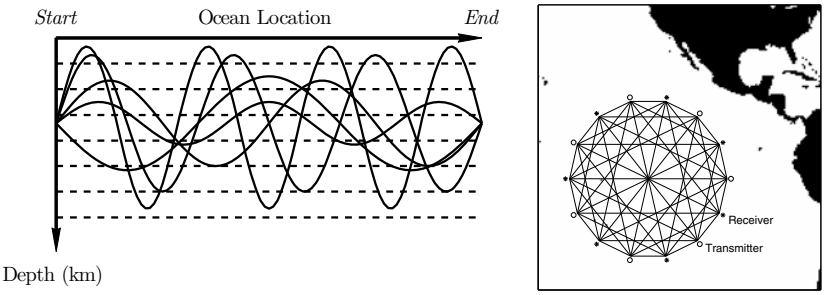


Fig. 2.10. Two inverse problems need to be solved. A 2D space–depth problem, left, inverts the acoustic arrival measurements to average sound speed by layer. A second 3D space–space–depth problem, right, solves for the three-dimensional pattern of sound speed based on many source–receiver pairs, in order to infer ocean temperature and currents.

2. **Invert the 3D tomographic problem:**
   As shown in Figure 2.10, we may have a number of source–receiver pairs, *each* of which leads to a 2D inverse problem in (2.99). We now have a three-dimensional inverse problem, with sources and receivers distributed spatially, and with layers over depth. Although such problems are widely studied in the medical imaging literature, the ocean acoustic problem is far more sparse and irregular, making many medical approaches inapplicable.

Summary

An inverse problem exists any time we wish to infer one or more unknowns from a set of measurements. The forward problem proceeds from the unknowns to the measurements; the inverse problem is the inference from the measurements to the unknowns. The inverse problem is said to be well-posed if the following three criteria are satisfied:

EXISTENCE: Every observation $\underline{m}$ has at least one corresponding value of $\underline{z}$.

UNIQUENESS: For every observation $\underline{m}$, the solution for $\underline{z}$ is unique.

CONTINUITY: The dependence of the solution $\underline{z}$ on $\underline{m}$ is continuous.

However, even problems which are well-posed may fail to be solvable numerically because of poor conditioning, in which case we *regularize* the problem:

$$\underline{\hat{z}}(\underline{m}, \lambda) = \arg_{\underline{z}} \min \left\{ \|\underline{m} - C\underline{z}\|_{R^{-1}} + \lambda \|L\underline{z}\| \right\}, \tag{2.100}$$

where λ is the regularization constant, which may be learned by validation, and where L is a set of assumptions or constraints on $\underline{z}$.

For Further Study

To look further into inverse problems, the papers by Bertero *et al* [27] and by Geman [129] are highly recommended, as they combine questions of inverse problems and low-level computer vision in the former paper, and random fields in the latter.

To read further in the area of regularization, three papers are recommended [27, 189, 304], in addition to the classic text [308] by Tikhonov and Arsenin.

The references, sorted by category starting on page 433, give additional text and paper citations.

Sample Problems

Problem 2.1: Uniqueness and Ill-Posed Problems

(a) Define "Uniqueness"
(b) Given matrix $C = \begin{bmatrix} 1 & 2 \end{bmatrix}$:
- Compute Rank(C)
- Compute or describe Ra(C), Nu(C)
- Does the inverse problem $m = C\underline{z}$ satisfy uniqueness?

(c) In general, for C a $k \times n$ matrix, prove that uniqueness fails for $k < n$.

Problem 2.2: Existence and Ill-Posed Problems

(a) Define "Existence"
(b) Given the matrix

$$C = \begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 1 & 1 \end{bmatrix}$$

- Compute Rank(C)
- Compute or describe Ra(C), Nu(C)
- Does the inverse problem $m = C\underline{z}$ satisfy existence?

(c) In general, for C an $k \times n$ matrix, prove that uniqueness fails for $k > n$.

Problem 2.3: Number of Solutions

For a linear inverse problem, prove that the number of solutions must either be zero, one, or infinitely many.

Problem 2.4: Numerical Conditioning

We want to look at matrix conditioning, as a function of statistics and matrix size. We imagine that we have a vector $\underline{x}$ having n elements, such that $\underline{x}$ has an associated $n \times n$ covariance P.

We can study the numerical behaviour of P in a variety of ways; here we consider two:

1. The condition number $\kappa(P)$, evaluated in MATLAB using function `cond`
2. The degree of error in matrix inversion. Let

$$e = \max\Big(\text{diag}\big(P \cdot P^{-1}\big)\Big) - \min\Big(\text{diag}\big(P \cdot P^{-1}\big)\Big).$$

In the ideal case, $P \cdot P^{-1} = I$, in which case $e = 0$. Larger values of e imply greater difficulties in computing accurate inverses, a surrogate for conditioning.

In a numerical mathematical package, such as MATLAB, prepare the following:

(a) Test conditioning as a function of the degree of correlation:
Create matrices $A_1, \ldots, A_{50}$ where

$$A_n \text{ is } 20 \times 20, \quad (A_n)_{i,j} = \exp(-|i-j| \cdot 0.3^n).$$

(b) Test conditioning as a function of matrix size:
Create exponentially-correlated matrices $B_1, \ldots, B_{50}$ where

$$B_n \text{ is } n \times n, \quad (B_n)_{i,j} = \exp\left(-\frac{|i-j|}{3}\right).$$

(c) As in (b), with the same correlation length (of 3), but with different statistics:
Create Gaussian-correlated matrices $C_1, \ldots, C_{50}$ where

$$C_n \text{ is } n \times n, \quad (C_n)_{i,j} = \exp\left(-\tfrac{1}{2}\left(\frac{i-j}{3}\right)^2\right).$$

Plot the condition number κ and inversion inconsistency e for each of A_n, B_n, C_n as a function of n. Study the behaviour of the plots.

Problem 2.5: Open-Ended Real-Data Problem — Image Processing

Most numerical mathematics programs offer a variety of regularization methods and inverse problems. Both MATLAB and OCTAVE, a free open-source alternative, offer most of the following routines:

`medfilt2`	Two-dimensional median filtering
`filter2`	Two-dimensional image filtering
`wiener2`	The two-dimensional Wiener filter for denoising
`iradon`	The inverse Radon transform, for medical tomography
`deconvwnr`	The two-dimensional Wiener filter for deblurring
`deconvreg`	Image deblurring using a regularized filter
`ipexregularized`	Image deblurring example

Acquire some number of images, of your own or from the Internet, and experiment with one or more of the above functions:

(a) For denoising, report your results as a function of the added noise variance, and the type of noise (Gaussian, uniform, salt-and-pepper).
(b) For deblurring, report your results as a function of the degree of blur.

3

Static Estimation and Sampling

This chapter derives the two fundamental linear estimators:

Section 3.1: The non-Bayesian linear least-squares estimator
Section 3.2: The Bayesian linear least-squares estimator

Throughout this chapter we concern ourselves with the derivation of algebraic estimators $\hat{\underline{z}}$ for some random vector $\underline{z}$, but ignoring the issues of what $\underline{z}$ represents, or any concerns regarding its size, both of which are extremely important and are examined closely beginning with Chapter 5.

The question of how to actually represent a multidimensional field in a vector is the topic of Chapter 5, and questions of computational tractability and simplification are addressed starting in Chapter 9.

In this chapter we consider $\underline{z}$ to be *static*; that is, where $\underline{z}$ is a single unknown vector which has no time dependence or evolution over time. Issues of time dependence are considered in Chapter 4.

Even within this static framework there are a number of variations, depending on what we assume or know about the measurement errors (and their statistics), and similarly whether $\underline{z}$ is just unknown, or subject to the constraints and assumption of some prior statistical model.

P. Fieguth, *Statistical Image Processing and Multidimensional Modeling*, Information Science and Statistics, DOI 10.1007/978-1-4419-7294-1_3, © Springer Science+Business Media, LLC 2011

3.1 Non-Bayesian Estimation

Linear Least Squares

We'll start with the simplest possible linear inverse problem, where we wish to infer unknowns $\underline{z}$ from measurements $\underline{m}$:

$$\underline{m} = C\underline{z}. \tag{3.1}$$

In many cases this problem will be ill-posed:

If $\underline{m} \in \text{Ra}(C)$ then existence is satisfied, but uniqueness may not be.
If $\underline{m} \notin \text{Ra}(C)$ then existence fails.
If $\text{Nu}(C) \neq \underline{0}$ then uniqueness fails.

Only if C is square and invertible is the problem well-posed.

The above pessimistic summary, that only problems in which C is invertible may be solved, is due to limitations in our problem formulation (3.1). We rarely expect C to be invertible; indeed, normally we have more measurements than unknowns, in which case C is a tall rectangular matrix. Furthermore we don't actually believe that $\underline{m} = C\underline{z}$; rather, measurements are invariably corrupted by some inaccuracy or noise $\underline{v}$,

$$\underline{m} = C\underline{z} + \underline{v}, \tag{3.2}$$

in which case we are interested in finding $\hat{\underline{z}}$ such that $C\hat{\underline{z}}$ is *close* to $\underline{m}$, but not necessarily equal. We often choose a least-squares criterion (2.90): if we define the estimation error $\underline{e}$ as

$$\underline{e} \triangleq \underline{m} - C\hat{\underline{z}}, \tag{3.3}$$

then the least-squares goal is to find $\hat{\underline{z}}$ by minimizing $\underline{e}^T\underline{e}$. There are many ways of deriving the solution to this problem; we consider two.

ALGEBRAIC DERIVATION: The squared error to minimize is

$$\underline{e}^T\underline{e} = (\underline{m} - C\hat{\underline{z}})^T(\underline{m} - C\hat{\underline{z}}), \tag{3.4}$$

thus the vector derivative (Appendix A.6) is

$$\frac{\partial \underline{e}^T\underline{e}}{\partial \hat{\underline{z}}} = \frac{\partial}{\partial \hat{\underline{z}}}\left[\underline{m}^T\underline{m} - 2\underline{m}^TC\hat{\underline{z}} + \hat{\underline{z}}^TC^TC\hat{\underline{z}}\right] \tag{3.5}$$

$$= \underline{0}^T - 2\underline{m}^TC + 2\hat{\underline{z}}^TC^TC \tag{3.6}$$

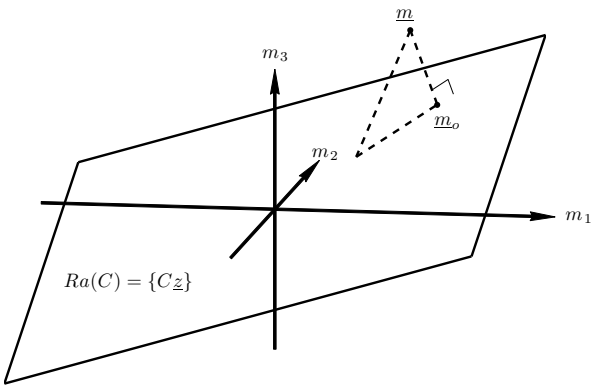


Fig. 3.1. The orthogonality principle: Existence fails here, since $\underline{m}$ lies outside of Ra(C). To solve this ill-posed problem, least-squares seeks that point in Ra(C) which is closest to $\underline{m}$. This closest point must be $\underline{m}_o$, the point where the error $\underline{m}-\underline{m}_o$ is at right angles (orthogonal) to Ra(C).

So $\underline{e}^T\underline{e}$ is minimized when $\hat{\underline{z}}^T C^T C = \underline{m}^T C$ or, taking the transpose, when $C^T C\hat{\underline{z}} = C^T\underline{m}$. In the event that $C^T C$ is invertible, meaning that C has full column rank, we obtain the well-known least-squares estimator[1]

$$\hat{\underline{z}} = \left(C^T C\right)^{-1} C^T \underline{m}. \tag{3.7}$$

GEOMETRIC DERIVATION — THE ORTHOGONALITY PRINCIPLE: We can reinterpret the problem in a more geometric sense. Since $C\hat{\underline{z}}$ can explore all of Ra(C), finding $\hat{\underline{z}}$ to minimize $\|\underline{m} - C\hat{\underline{z}}\|$ is equivalent to finding the element in Ra(C) closest to $\underline{m}$. That is, which vector $\bar{\underline{m}} = C\hat{\underline{z}} \in$ Ra(C) minimizes

$$\|\underline{m} - C\hat{\underline{z}}\| = \|\underline{m} - \bar{\underline{m}}\| = (\underline{m} - \bar{\underline{m}})^T(\underline{m} - \bar{\underline{m}})? \tag{3.8}$$

Let $\underline{m}_o$ be the unique element of Ra(C) such that the error $\underline{m} - \underline{m}_o$ is *orthogonal* to Ra(C); that is, that

$$(\underline{m} - \underline{m}_o) \perp \text{Ra}(C) \quad \leftrightarrow \quad (\underline{m} - \underline{m}_o)^T\underline{q} = 0 \ \ \forall \underline{q} \in \text{Ra}(C) \tag{3.9}$$

This construction decomposes the error

$$\begin{array}{ccccc} (\underline{m} - C\hat{\underline{z}}) & = & (\underline{m} - \underline{m}_o) & + & (\underline{m}_o - C\hat{\underline{z}}) \\ \underline{e} & = & \underline{e}_1 & + & \underline{e}_2 \end{array} \tag{3.10}$$

[1] It should be mentioned that the estimator in (3.7), although correct, should not actually be solved using an explicit matrix inverse, especially for large problems. This is discussed much later, in Section 9.1.

into two orthogonal components $\underline{e}_1 \perp \underline{e}_2$, such that by Pythagoras

$$\underbrace{\|\underline{m} - C\underline{\hat{z}}\|}_{\text{Error to Minimize}} = \underbrace{\|\underline{m} - \underline{m}_o\|}_{\text{Fixed}} + \underbrace{\|\underline{m}_o - C\underline{\hat{z}}\|}_{\text{Set to Zero}} . \tag{3.11}$$

Of the two terms on the right-hand side, the first is not under our control, so the best we can do is to set the second term to zero, which is accomplished by setting $C\underline{\hat{z}} = \underline{m}_o$.

Finally, how do we find $\underline{m}_o$? The projection $\underline{m}_o$ must satisfy two criteria:

1. $\underline{m}_o$ must be in the subspace Ra(C)
2. The difference $\underline{m} - \underline{m}_o$ must be orthogonal to Ra(C).

We claim that

$$\underline{m}_o = C\,(C^T C)^{-1} C^T \underline{m} \tag{3.12}$$

is the projection of $\underline{m}$ onto the range space Ra(C), which can be verified:

1. $\underline{m}_o = C\,(C^T C)^{-1} C^T \underline{m} = C[\cdots] \in \text{Ra}(C)$,
2. $C^T(\underline{m} - \underline{m}_o) = C^T \underline{m} - (C^T C)(C^T C)^{-1} C^T \underline{m} = \underline{0}$.

So we have identified $\underline{m}_o = C\,(C^T C)^{-1} C^T \underline{m}$ as the optimum element in the subspace of C, minimizing the error with respect to $\underline{m}$; thus we identify

$$C\underline{\hat{z}} = \underline{m}_o = C\,(C^T C)^{-1} C^T \underline{m} \quad \rightarrow \quad \underline{\hat{z}} = (C^T C)^{-1} C^T \underline{m} \tag{3.13}$$

as the selected estimator.

The computation of $\underline{\hat{z}} = (C^T C)^{-1} C^T \underline{m}$ requires the invertibility of $(C^T C)$, which is true if and only if C has full-column rank. This assumption may fail under the following circumstances:

- Suppose C is $k \times n, k < n$:
  We have fewer equations than unknowns and $(C^T C)$ is singular. The solution for $\underline{\hat{z}}$ is not unique, and additional constraints (prior model, regularization constraints) are required.
- Suppose C is $k \times n, k \geq n$:
  We have at least as many equations as unknowns; $(C^T C)$ will be singular if the columns of C are linearly dependent, meaning that some portion of $\underline{z}$ is unobserved (C has a nullspace).

Much of the above mathematics is represented elegantly in the theory of pseudoinverse matrices [4], which is summarized in Appendix A.9. In particular if C is nonsingular then the pseudoinverse C^+ of C is the matrix inverse $C^+ = C^{-1}$, generalizing to the projection operator $C^+ = (C^T C)^{-1} C^T$ in the special case that C has

full column rank. However unlike the projector, C^+ exists for *all* real C, whether rank deficient or not.

Indeed, given observation C, the estimator

$$\underline{\hat{z}} = C^+ \underline{m} \tag{3.14}$$

based on the Moore–Penrose pseudoinverse C^+ [4], produces the minimum-norm least-squares solution for $\underline{z}$. That is, of all of the (possibly many) $\underline{z}$ minimizing

$$(\underline{m} - C\underline{z})^T (\underline{m} - C\underline{z}), \tag{3.15}$$

$C^+ \underline{m}$ chooses that unique vector which minimizes $\underline{z}^T \underline{z}$.

Weighted Least Squares

Our previous derivations considered all of the measurement errors as equivalent, equally penalizing the error in each dimension. In practice we may prefer a weighted criterion, minimizing the least-squares weighted error $(F\underline{e})$:

$$(F\underline{e})^T (F\underline{e}) = \underline{e}^T W \underline{e} \equiv \|\underline{m} - C\underline{\hat{z}}\|_W \tag{3.22}$$

for some appropriate choice of $W = F^T F$. However, if we transform the error by weight matrix F,

$$\begin{array}{lll} \underline{e} = \underline{m} - C\underline{\hat{z}} & \Longrightarrow & F\underline{e} = F\underline{m} - FC\underline{\hat{z}} \\ & \Longrightarrow & \bar{\underline{e}} = \bar{\underline{m}} - \bar{C}\underline{\hat{z}} \end{array} \tag{3.23}$$

As a consequence, the error criterion (3.22) can be similarly transformed as

$$\underline{e}^T W \underline{e} = (F\underline{e})^T (F\underline{e}) = \bar{\underline{e}}^T \bar{\underline{e}}. \tag{3.24}$$

That is, we have converted the problem to regular least squares, to which we have already derived the answer in (3.7),(3.13):

$$\underline{\hat{z}} = (\bar{C}^T \bar{C})^{-1} \bar{C}^T \bar{\underline{m}} \tag{3.25}$$

$$= \left(C^T F^T \cdot FC\right)^{-1} (C^T F^T) \cdot F\underline{m} = (C^T W C)^{-1} C^T W \underline{m}, \tag{3.26}$$

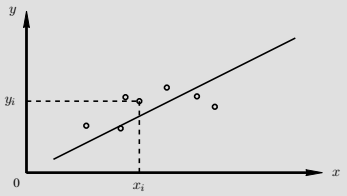
where if W is nonsingular, the conditions for the invertibility of $(C^T W C)$ are the same as those for $(C^T C)$.

Constrained or Regularized Least-Squares

The reader may wonder regarding the relevance of the discussions of least-squares solutions to $\underline{m} = C\underline{z} + \underline{v}$, given that much of Chapter 2 was devoted to demonstrating the ill-posedness of most such problems and, in Section 2.4, the discussion

Example 3.1: Linear Regression is Least Squares

Suppose we are given points $\{x_i, y_i\}$:



We wish to do linear regression, meaning that we assert a model

$$y_i = ax_i + b + v_i, \tag{3.16}$$

where v_i is some error. Thus we have unknowns $\underline{z}$ and measurements $\underline{m}$:

$$\underline{z} = \begin{bmatrix} a \\ b \end{bmatrix} \qquad \underline{m} = \begin{bmatrix} y_1 \\ \vdots \\ y_k \end{bmatrix} \tag{3.17}$$

leading to the usual formulation

$$\underline{m} = \begin{bmatrix} y_1 \\ \vdots \\ y_k \end{bmatrix} = \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} + \underline{v} \equiv C\underline{z} + \underline{v}. \tag{3.18}$$

We know the usual least-squares solution to this canonical problem to be

$$\hat{\underline{z}} = \left(C^T C\right)^{-1} C^T \underline{m}. \tag{3.19}$$

Substituting (3.18) into (3.19) gives

$$\hat{\underline{z}} = \begin{bmatrix} \hat{a} \\ \hat{b} \end{bmatrix} = \begin{bmatrix} \sum x_i^2 & \sum x_i \\ \sum x_i & \sum 1 \end{bmatrix}^{-1} \begin{bmatrix} \sum x_i y_i \\ \sum y_i \end{bmatrix} \tag{3.20}$$

which is the familiar solution to linear regression.

Once we recognize and understand the above development, linear regression is easily generalized to other cases, such as

$$y_i = ax_i^2 + bx_i + v_i \qquad y_i = ae^{x_i} + v_i \qquad y_i = a\sin x_i + b\cos 2x_i + v_i \tag{3.21}$$

y_i does *not* need to be linear in x_i, however it *does* need to be linear in the unknowns a, b. For example, the generalized regression $y_i = \sin(ax_i) + v_i$ is a much more difficult problem.

of constraints and regularization schemes to permit a solution. In particular, (2.39) formulated the following constrained least-squares criterion:

$$\min\left\{\|\underline{m} - C\underline{\hat{z}}\|_{R^{-1}} + \lambda\|L\underline{\hat{z}}\|\right\}. \tag{3.27}$$

This criterion is, however, just a variant of weighted least-squares:

$$\begin{aligned}
&\|\underline{m} - C\underline{\hat{z}}\|_{R^{-1}} + \lambda\|L\underline{\hat{z}}\| \\
&\quad = (\underline{m} - C\underline{\hat{z}})^T R^{-1}(\underline{m} - C\underline{\hat{z}}) + \lambda(L\underline{\hat{z}})^T(L\underline{\hat{z}}) \\
&\quad = \begin{bmatrix}\underline{m} - C\underline{\hat{z}} \\ \sqrt{\lambda}L\underline{\hat{z}}\end{bmatrix}^T \begin{bmatrix}R^{-1} & \\ & I\end{bmatrix} \begin{bmatrix}\underline{m} - C\underline{\hat{z}} \\ \sqrt{\lambda}L\underline{\hat{z}}\end{bmatrix} \\
&\quad = \left(\begin{bmatrix}\underline{m} \\ \underline{0}\end{bmatrix} - \begin{bmatrix}C \\ -\sqrt{\lambda}L\end{bmatrix}\underline{\hat{z}}\right)^T \begin{bmatrix}R^{-1} & \\ & I\end{bmatrix} \left(\begin{bmatrix}\underline{m} \\ \underline{0}\end{bmatrix} - \begin{bmatrix}C \\ -\sqrt{\lambda}L\end{bmatrix}\underline{\hat{z}}\right) \\
&\quad = \left(\underline{\bar{m}} - \bar{C}\underline{\hat{z}}\right)^T \bar{R}^{-1}\left(\underline{\bar{m}} - \bar{C}\underline{\hat{z}}\right) \\
&\quad = \left\|\underline{\bar{m}} - \bar{C}\underline{\hat{z}}\right\|_{\bar{R}^{-1}}
\end{aligned} \tag{3.28}$$

That is, the regularization constraints $\lambda\|L\underline{\hat{z}}\|$ are algebraically equivalent to the inclusion of an additional "measurement"

$$L\underline{z} = \underline{0}. \tag{3.29}$$

Associating the above criterion $\left\|\underline{\bar{m}} - \bar{C}\underline{\hat{z}}\right\|_{\bar{R}^{-1}} = \underline{e}^T\bar{R}^{-1}\underline{e}$ with the criterion $\underline{e}^T W\underline{e}$ from weighted least squares allows us to find the solution directly from the weighted least-squares solution (3.26), with $W = \bar{R}^{-1}$:

$$\underline{\hat{z}} = (\bar{C}^T\bar{R}^{-1}\bar{C})^{-1}\bar{C}^T\bar{R}^{-1}\underline{\bar{m}} \tag{3.30}$$

$$= \left(\begin{bmatrix}C \\ -\sqrt{\lambda}L\end{bmatrix}^T \begin{bmatrix}R^{-1} & \\ & I\end{bmatrix} \begin{bmatrix}C \\ -\sqrt{\lambda}L\end{bmatrix}\right)^{-1} \begin{bmatrix}C \\ -\sqrt{\lambda}L\end{bmatrix}^T \begin{bmatrix}R^{-1} & \\ & I\end{bmatrix} \begin{bmatrix}\underline{m} \\ \underline{0}\end{bmatrix} \tag{3.31}$$

$$= \left(C^T R^{-1}C + \lambda L^T L\right)^{-1} C^T R^{-1}\underline{m} \tag{3.32}$$

This result, the estimator for regularized least-squares, is of key importance throughout the following chapters.

Maximum Likelihood

A related approach is that of *maximum likelihood*, still treating $\underline{z}$ as unknown (i.e., non-Bayesian), but where we formulate the problem more explicitly using statistics. Given measurements $\underline{m}$ with known Gaussian noise statistics

$$\underline{m} = C\underline{z} + \underline{v} \qquad \underline{v} \sim \mathcal{N}(\underline{0}, R). \tag{3.33}$$

we seek the maximum likelihood estimate (2.93)

$$\hat{\underline{z}}_{ML} \triangleq \arg_{\underline{z}} \max\, p(\underline{m}|\underline{z}) \tag{3.34}$$

$$= \arg_{\underline{z}} \max\, \frac{1}{(2\pi)^{n/2}|R|} \exp\left(-\frac{1}{2}(\underline{m} - C\underline{z})^T R^{-1}(\underline{m} - C\underline{z})\right) \tag{3.35}$$

$$= \arg_{\underline{z}} \min\, (\underline{m} - C\underline{z})^T R^{-1}(\underline{m} - C\underline{z}) \tag{3.36}$$

$$= \arg_{\underline{z}} \min\, (\underline{e})^T R^{-1}(\underline{e}) \tag{3.37}$$

which we immediately recognize as the weighted least-squares criterion (3.22), where we identify $W = R^{-1}$. The solution follows from (3.26)

$$\hat{\underline{z}} = (C^T R^{-1} C)^{-1} C^T R^{-1} \underline{m}. \tag{3.38}$$

Associating W with R^{-1} is intuitive. A *small* value R_{ii} implies a small measurement noise variance v_i, meaning that we expect to find an estimate $\hat{\underline{z}}$ which makes e_i small. Therefore we wish to assert an intolerance for large e_i, implying a large W_{ii}.

Thus we have derived the LLS, WLS, and ML estimators, and have seen the great deal which they have in common. It is important to keep in mind that the parallels with ML do assume Gaussianity, and do not hold for other statistics.

3.2 Bayesian Estimation

We move now to *Bayesian* estimation, in which our unknowns $\underline{z}$ are *random* — statistical quantities subject to a prior model.

The Bayesian formulation leads to a few subtle changes, compared to the non-Bayesian case. The error $\underline{e}$ is now the estimation error

$$\underline{e} = \tilde{\underline{z}} = \hat{\underline{z}} - \underline{z}, \tag{3.39}$$

and the stochastic nature of $\underline{z}$ implies that $\underline{e}$ too is stochastic, so the least-squares criterion is formulated as an expectation:

$$\hat{\underline{z}} = \arg_{\hat{\underline{z}}} \min\, E\left[(\hat{\underline{z}} - \underline{z})^T (\hat{\underline{z}} - \underline{z}) | \underline{m}\right]. \tag{3.40}$$

The orthogonality principle still applies, but now in a statistical form, rather than strictly geometrical. That is, orthogonality states that the optimum estimator must make the error uncorrelated (orthogonal) to any function of the measurements:

$$E\left[(\hat{\underline{z}} - \underline{z}) f(\underline{m})^T\right] = \mathbf{0} \qquad \text{for } \textit{any} \text{ function } f. \tag{3.41}$$

This makes sense: if there were any residual correlation remaining between the measurements and the estimation error, then we should be able to formulate a *better* estimator by taking advantage of this correlation.

Bayesian Least Squares

The general Bayes least-squares estimator (2.79) is easily derived. For simplicity and clarity, we derive the estimator for the scalar case:

$$\hat{z} \triangleq \arg_{\hat{z}} \min \left\{ E[(\hat{z} - z)^2 | m] \right\} \tag{3.42}$$

$$= \arg_{\hat{z}} \min \int (\hat{z} - z)^2 p(z|m) dz. \tag{3.43}$$

To minimize this expression we set the derivative to zero:

$$\frac{\partial}{\partial \hat{z}} \int (\hat{z} - z)^2 p(z|m) dz = \int 2(\hat{z} - z) p(z|m) dz = 0 \tag{3.44}$$

$$\underbrace{\int \hat{z} p(z|m) dz}_{\hat{z}} = \underbrace{\int z p(z|m) dz}_{E[z|m]} \tag{3.45}$$

That is, the optimal estimator is just the conditional mean of z, given the measurements m. The result for the vector case is identical:

$$\underline{\hat{z}} = E[\underline{z}|\underline{m}] \tag{3.46}$$

This simple result is the consequence of the quadratic (least-squares) criterion, which simplifies when differentiated. However, the simplicity of this estimator (3.46) is deceiving: the conditional mean of $\underline{z}$ may be an extremely complicated, nonlinear function of $\underline{m}$, motivating the development of the simpler Bayesian *linear* least-squares estimator, below.

We conclude with three properties of the Bayesian estimator.

- **Bias:** The Bayesian estimator is always unbiased:

$$\underline{b} = E[\underline{e}|\underline{m}] = E[\underline{\hat{z}} - \underline{z}|\underline{m}] = \underline{\hat{z}} - E[\underline{z}|\underline{m}] = \underline{0}. \tag{3.47}$$

- **Orthogonality:** The estimation error is orthogonal to (uncorrelated with) *any* function $f()$ of the measurements:

$$E\left[\underline{e} \cdot f(\underline{m})^T\right] = E\left[(\underline{\hat{z}} - \underline{z}) \cdot f(\underline{m})^T\right] = \mathbf{0}. \tag{3.48}$$

- **Linearity:** If the measurements $\underline{m}$ and the unknowns $\underline{z}$ are jointly Gaussian, then the estimator $\underline{\hat{z}}$ is a linear function of $\underline{m}$.

Bayesian Linear Least Squares

Because of the complexity in evaluating the expectation (3.46) in the general Bayesian estimator for very large estimation problems we are motivated to find a simpler approach. In particular, we would like a *linear* estimator which can be expressed in explicit, closed form.

The Bayesian linear least-squares estimator (LLSE) is that linear estimator which, of all possible linear estimators, minimizes the squared error. Given that the estimator is linear, it must be of the general form

$$\hat{\underline{z}} = A\underline{m} + \underline{\alpha}. \tag{3.49}$$

We derive $A, \underline{\alpha}$ by asserting the following two criteria:

1. Unbiasedness — the estimator $\hat{\underline{z}}$ is unbiased:
$$E\left[\hat{\underline{z}} - \underline{z}\right] = \underline{0}. \tag{3.50}$$

2. Orthogonality — the error in $\hat{\underline{z}}$ must be uncorrelated (orthogonal) with any *linear* function of the data:
$$E\left[(\hat{\underline{z}} - \underline{z})(F\underline{m} + \underline{g})^T\right] = E\left[(\hat{\underline{z}} - \underline{z})\right] E\left[(F\underline{m} + \underline{g})^T\right] \qquad \forall F, \underline{g}. \tag{3.51}$$

Asserting these two criteria, we derive the constraints on F and $\underline{g}$:

1. Unbiasedness:
$$E\left[\hat{\underline{z}} - \underline{z}\right] = E\left[A\underline{m} + \underline{\alpha} - \underline{z}\right] = A\underline{\mu}_m + \underline{\alpha} - \underline{\mu}_z = \underline{0} \tag{3.52}$$
therefore
$$\Longrightarrow \underline{\alpha} = \underline{\mu}_z - A\underline{\mu}_m. \tag{3.53}$$

2. Orthogonality: from the unbiasedness condition of (3.50)
$$E\big[(\hat{\underline{z}} - \underline{z})\big] E\big[(F\underline{m} + \underline{g})^T\big] = \mathbf{0}. \tag{3.54}$$
We can therefore simplify the left-hand side of the orthogonality condition (3.51) as
$$\begin{aligned} \mathbf{0} = E\big[(\hat{\underline{z}} - \underline{z})(F\underline{m} + \underline{g})^T\big] \\ &= E\big[(A\underline{m} + \underline{\mu}_z - A\underline{\mu}_m - \underline{z})(F\underline{m} + \underline{g})^T\big] \\ &= AE\big[(\underline{m} - \underline{\mu}_m)\underline{m}^T\big]F^T + AE\big[(\underline{m} - \underline{\mu}_m)\big]\underline{g}^T \\ &\quad - E\big[(\underline{z} - \underline{\mu}_z)\underline{m}^T\big]F^T - E\big[(\underline{z} - \underline{\mu}_z)\big]\underline{g}^T \\ &= A\Lambda_m F^T + A \cdot \underline{0} \cdot \underline{g}^T - \Lambda_{zm}F^T - \underline{0} \cdot \underline{g}^T, \end{aligned} \tag{3.55}$$

where Λ denotes a covariance or cross-covariance

$$\Lambda_m = \text{cov}(\underline{m}) \qquad \Lambda_{zm} = E\big[(\underline{z} - \underline{\mu}_z)(\underline{m} - \underline{\mu}_m)^T\big]. \tag{3.56}$$

Thus it is required that

$$(A\Lambda_m - \Lambda_{zm})F^T = \mathbf{0} \qquad \forall F. \tag{3.57}$$

For this expression to be valid for *all possible* F, it follows that

$$A\Lambda_m - \Lambda_{zm} = \mathbf{0} \quad \Longrightarrow \quad A = \Lambda_{zm}\Lambda_m^{-1}. \tag{3.58}$$

The derived LLSE estimator is then the well known formula

$$\hat{\underline{z}} = A\underline{m} + \underline{\alpha} = \underline{\mu}_z + \Lambda_{zm}\Lambda_m^{-1}(\underline{m} - \underline{\mu}_m). \tag{3.59}$$

This estimator is actually quite intuitive: our estimate for $\hat{\underline{z}}$ is given by the mean $\underline{\mu}_z$ plus an offset due to the "surprise" in the measurements: the difference $(\underline{m} - \underline{\mu}_m)$ between what was observed and what was expected, normalized by the expected random (noisy) variations Λ_m in the measurements, and unit-converted by Λ_{zm} from units of $\underline{m}$ to those of $\underline{z}$.

We can also derive the error statistics for the estimator:

$$\begin{aligned}
&\text{cov}(\hat{\underline{z}} - \underline{z}) \\
&\quad = \text{cov}\left(\Lambda_{zm}\Lambda_m^{-1}(\underline{m} - \underline{\mu}_m) - (\underline{z} - \underline{\mu}_z)\right) &(3.60)\\
&\quad = (\Lambda_{zm}\Lambda_m^{-1})\Lambda_m(\Lambda_{zm}\Lambda_m^{-1})^T - (\Lambda_{zm}\Lambda_m^{-1})\Lambda_{mz} - \Lambda_{zm}(\Lambda_{zm}\Lambda_m^{-1})^T + \Lambda_z &(3.61)
\end{aligned}$$

Thus the error covariance $\text{cov}(\tilde{\underline{z}})$ is given by the *prior* covariance Λ_z, minus a reduction controlled by the quality and degree of relevance of the measurements:

$$\text{cov}(\tilde{\underline{z}}) = \Lambda_z - \Lambda_{zm}\Lambda_m^{-1}\Lambda_{zm}^T. \tag{3.62}$$

The estimator offers no or limited improvement,

$$\text{cov}(\tilde{\underline{z}}) \simeq \Lambda_z, \tag{3.63}$$

if the measurements are very noisy (Λ_m large) or if the measurements are irrelevant by being uncorrelated with the unknowns ($\Lambda_{zm} \simeq \mathbf{0}$).

3.2.1 Bayesian Static Problem

The previous section derived the LLSE, but in terms of second-order statistics and not in terms of the parameters of the static problem.

Given the static problem from Section 2.5.1:

$$\underline{z} \sim \mathcal{N}(\underline{\mu}, P) \qquad \underline{m} = C\underline{z} + \underline{v} \qquad \underline{v} \sim \mathcal{N}(\underline{0}, R), \tag{3.64}$$

we can easily derive the relevant second-order statistics:

$$
\begin{aligned}
\underline{\mu}_z &= \underline{\mu} \\
\underline{\mu}_m &= E\big[C\underline{z} + \underline{v}\big] = C\underline{\mu}_z + \underline{0} = C\underline{\mu} \\
\Lambda_z &= P \\
\Lambda_{zm} &= E\big[(\underline{z} - \underline{\mu}_z)(\underline{m} - \underline{\mu}_m)^T\big] = E\big[(\underline{z} - \underline{\mu})(C\underline{z} + \underline{v} - C\underline{\mu})^T\big] = PC^T \\
\Lambda_m &= \mathrm{cov}(\underline{m} - \underline{\mu}_m) = \mathrm{cov}\big(C(\underline{z} - \underline{\mu}) + \underline{v}\big) = C\Lambda_z C^T + R
\end{aligned}
$$

Inserting these statistics into the LLSE (3.59), (3.62) we find **Form I** of the static estimator:

$$\hat{\underline{z}}(\underline{m}) = \underline{\mu} + PC^T(CPC^T + R)^{-1}(\underline{m} - C\underline{\mu}) \tag{3.65}$$

$$\tilde{P} = \mathrm{cov}(\tilde{\underline{z}}) = P - PC^T(CPC^T + R)^{-1}CP \tag{3.66}$$

By manipulating the above equations (using the ABCD lemma, (A.39)), there exists a second algebraically equivalent variant, **Form II**:

$$\hat{\underline{z}}(\underline{m}) = \underline{\mu} + \left(C^T R^{-1} C + P^{-1}\right)^{-1} C^T R^{-1}(\underline{m} - C\underline{\mu}) \tag{3.67}$$

$$\tilde{P} = \mathrm{cov}(\tilde{\underline{z}}) = \left(C^T R^{-1} C + P^{-1}\right)^{-1} \tag{3.68}$$

The equivalence of these two forms is by no means obvious or intuitive, and the proof of the ABCD lemma, connecting these two forms, requires some effort. However, a number of comments on these two forms are appropriate, as the suitability of one form or the other will vary with context:

Form and Derivation
: The structure of Form I is clearly linked to, and directly derived from, the LLSE, whereas Form II is structurally quite similar to regularized least-squares (3.32).

Model Assumptions
: Form I is explicitly expressed in terms of P and R, and the invertibility of P and R is not assumed, thus this estimator may be appropriate for certain singular estimation problems.

  On the other hand, Form II is expressed in terms of P^{-1} and R^{-1}. We show in Chapters 5 and 6 that in many cases a statistical model may directly specify sparse P^{-1}, rather than P.

Computational Complexity
: In Form I, the matrix inversion has the dimensions of $\underline{m}$, the measurements, whereas the inversion in Form II has the dimensions of $\underline{z}$, the state. In those cases where the number of measurements and unknowns is significantly different, the choice of form may be a significant factor in determining complexity.

3.2.2 Bayesian Estimation and Prior Means

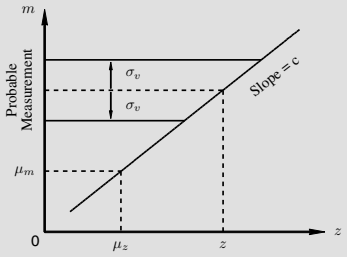
One final comment is in order before we complete our discussion of static estimation. All of the derived Bayesian estimators exhibit a separation between the *deterministic*

Example 3.2: Simple Scalar Estimation

A reader, unfamiliar with Bayesian estimation, may find it useful to consider a simple scalar problem. Suppose we have the following prior and measurement models:

$$z \sim (\mu_z, \sigma_z^2) \qquad m = cz + v \qquad v \sim (0, \sigma_v^2). \tag{3.69}$$

The forward model then projects the unknown value z to measurement m:



From (3.67) we know the solution to the inverse problem to be

$$\hat{z} = \mu_z + \frac{\tilde{p}c}{\sigma_v^2}(m - c\mu_z) = \mu_z + k(m - \mu_m) \tag{3.70}$$

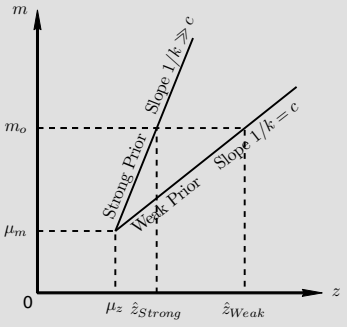
where, from (3.68), the estimator gain k can be evaluated as

$$k = \frac{\tilde{p}c}{\sigma_v^2} = \left(1/\sigma_z^2 + c^2/\sigma_v^2\right)^{-1} c/\sigma_v^2. \tag{3.71}$$

The estimated value $\hat{z}$ will therefore be a function of the degree to which the prior is asserted:

$$\begin{array}{rlll} \text{Weak Prior } (\sigma_z \text{ large}) \longrightarrow & \tilde{p} \simeq \sigma_v^2/c^2 & k \simeq 1/c & \hat{z} \simeq m/c \\ \text{Strong Prior } (\sigma_z \text{ small}) \longrightarrow & \tilde{p} \ll \sigma_v^2/c^2 & k \ll 1/c & \hat{z} \to \mu_z \end{array}$$

as drawn below:



and *stochastic* aspects of an estimation problem, where the deterministic portions are those aspects known *a priori* (such as the mean).

We can define a mean-removed estimation problem

$$\underline{z}' \triangleq \underline{z} - \underline{\mu} \qquad \underline{m}' = \underline{m} - C\underline{\mu} \tag{3.72}$$

which leads to the simplified (zero-mean) version of (3.67)

$$\underline{\hat{z}}'(\underline{m}) = \left(C^T R^{-1} C + P^{-1}\right)^{-1} C^T R^{-1} \underline{m}'. \tag{3.73}$$

By inspection, it is clear that we can recover the original, mean-included estimates just by adding back the mean:

$$\underline{\hat{z}} = \underline{\hat{z}}' + \underline{\mu}. \tag{3.74}$$

Since the original estimator (3.67) wasn't actually that complicated, why is this mean removal of interest?

1. It highlights and clarifies the important distinction between the deterministic and stochastic portions of an estimation problem, and that these two parts can really be processed separately.
2. In the dynamic estimation context of Chapter 4 the deterministic–stochastic separation still holds, however the equations become more complicated than in the static case, and so the simplification offered by a zero-mean assumption may be significant.
3. In many problems of practical interest, the definition of "the mean" may be very ambiguous. In computing estimates of ocean temperature, per Example 3.3, is "the mean" an average over one day, over one year, over one century? In principle the ocean temperature is ever-changing, and probably obeys no mean. In practice, however, we are normally interested in producing estimates over a certain time-scale, so by removing the behaviours which vary over much *longer* time-scales we relieve ourselves of the burden of modelling those behaviours, leaving us with a simpler problem.

We therefore adopt a relaxed policy with regard to knowing or specifying the mean in derivations and examples, with the understanding that a mean term can always be added to the estimates as a separate step.

3.2.3 Approximate Bayesian Estimators

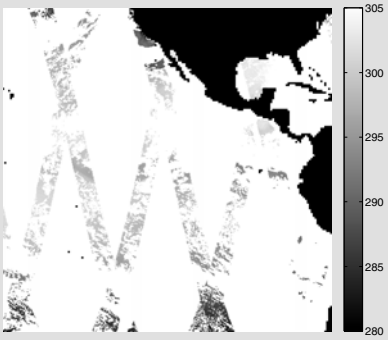
As we are interested, throughout this book, in computationally efficient, *approximate* estimators, it is worth considering the effects of an approximation.

Example 3.3: Prior Mean Removal

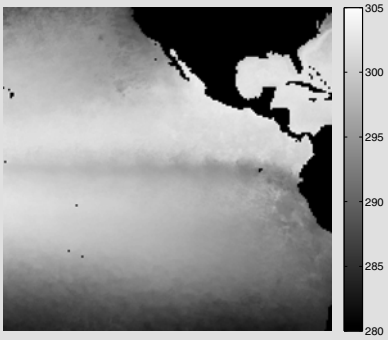
Suppose we wish to produce a map of ocean surface temperature.

We start, for example, with the three days of ATSR data shown here. The temperature is difficult to model, because the observed temperature is based on the superposition of at least two effects:

1. Short-term patterns, such as storms and cold fronts.
2. Long-term patterns, for example that the tropics tend to be warm, with the oceans cooling towards the poles.

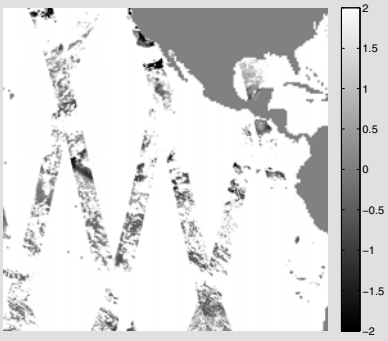


If we wish to study short-term temperature fluctuations, then the long-term variations are a nuisance: they are an additional phenomenon to model, and furthermore the large temperature range between poles and tropics means that subtle variations may be masked.



Based on an entire year of temperature data, we can produce an approximate sea-surface temperature mean, as shown here. The image represents phenomena unchanging over the period of a year: the warm Gulf of Mexico, the cool water of the South Equatorial Current etc.

Returning to the three days of ATSR data from the top figure, if we subtract out the estimated mean, then we arrive at a very different dataset, as shown. The dynamic range of the data is much smaller (4 degrees here, 25 degrees above), and subtle warming and cooling variations can be seen, both in the tropics and elsewhere. The statistical behaviour of these mean-removed data (3.72) is much simpler than before; at this point we learn a model for $(\underline{z} - \underline{\mu})$ and proceed.



Suppose that we have a zero-mean random vector $\underline{z}$; then from (3.65)–(3.68) we know the forms of the optimum linear least-squares estimator.

For any of a number of reasons, the matrices P, R may be difficult to invert or store in memory, and so we may wish to consider solving a very similar problem which yields similar (i.e., approximated) results:

$$\bar{P} \simeq P, \quad \bar{R} \simeq R \quad \Longrightarrow \quad \hat{\tilde{\underline{z}}} \simeq \hat{\underline{z}}, \quad \tilde{\bar{P}} \simeq \tilde{P}. \tag{3.75}$$

Whether the approximation is good or not is a difficult question, to be discussed in later chapters. However, in assessing an approximation it is important that we clearly distinguish three different error covariances:

$\tilde{P} = \text{cov}(\hat{\underline{z}} - \underline{z})$ The error covariance of the *optimum* estimator.

$\tilde{\bar{P}} \simeq \text{cov}(\hat{\underline{z}} - \underline{z})$ The error covariance *computed by* the approximate estimator.

$\tilde{P}_{\bar{z}} = \text{cov}(\hat{\tilde{\underline{z}}} - \underline{z})$ The *actual* error covariance of the approximate estimator.

The optimality of the original estimator clearly requires that

$$\tilde{P} \leq \tilde{P}_{\bar{z}}, \tag{3.76}$$

however the relationship between $\tilde{\bar{P}}$ and $\tilde{P}, \tilde{P}_{\bar{z}}$ is much less clear. For example, the particular choices of $\bar{P}, \bar{R}$ might lead to a naïvely optimistic estimator, $\tilde{\bar{P}} < \tilde{P}$, or an unnecessarily pessimistic one, $\tilde{\bar{P}} > \tilde{P}_{\bar{z}}$.

That is, in assessing the approximate estimator it is crucial to distinguish between what *it thinks* its covariance is ($\tilde{\bar{P}}$), and its *actual* covariance ($\tilde{P}_{\bar{z}}$).

Suppose we are given the standard static estimation problem

$$\underline{z} \sim \mathcal{N}(\underline{0}, P) \qquad \underline{m} = C\underline{z} + \underline{v} \qquad \underline{v} \sim \mathcal{N}(\underline{0}, R) \tag{3.77}$$

and we select an approximate estimator

$$\hat{\tilde{\underline{z}}} = \bar{K}\underline{m} \simeq K\underline{m}. \tag{3.78}$$

Then the *actual* error covariance of this estimator is

$$\begin{aligned} \tilde{P}_{\bar{z}} &= \text{cov}(\hat{\tilde{\underline{z}}} - \underline{z}) = \text{cov}(\bar{K}\underline{m} - \underline{z}) && (3.79)\\ &= \text{cov}(\bar{K}C\underline{z} + \bar{K}\underline{v} - \underline{z}) && (3.80)\\ &= \text{cov}\big((\bar{K}C - I)\underline{z} + \bar{K}\underline{v}\big) && (3.81)\\ &= (\bar{K}C - I)P(\bar{K}C - I)^T + \bar{K}R\bar{K}^T. && (3.82) \end{aligned}$$

Although it looks a bit cumbersome, this latter expression (3.82) is an important result, allowing us to find the actual error covariance for *any* linear estimator.

3.2.4 Bayesian / NonBayesian Duality

Philosophically, the Bayesian and non-Bayesian perspectives lie at opposite ends of a spectrum, and there has been vigorous debate as to whether prior knowledge is of utmost importance, on the one hand, or statistically irrelevant, on the other.

It is rather surprising, then, that a comparison of the Bayesian estimator (3.67) and the non-Bayesian least-squares estimators (3.26),(3.32) reveals an astonishing degree of similarity, despite very different formulations and optimization objectives:

Weighted Least-Squares:

$$\begin{aligned}&\text{Minimize } \|\underline{m} - C\underline{\hat{z}}\|_W \\ &\Longrightarrow \underline{\hat{z}}(\underline{m}) = \left(C^T W C\right)^{-1} C^T W \underline{m}\end{aligned}$$

Regularized Least-Squares:

$$\begin{aligned}&\text{Minimize } \|\underline{m} - C\underline{\hat{z}}\|_{R^{-1}} + \lambda \|L\underline{\hat{z}}\| \\ &\Longrightarrow \underline{\hat{z}}(\underline{m}) = \left(\lambda L^T L + C^T R^{-1} C\right)^{-1} C^T R^{-1} \underline{m}\end{aligned}$$

Bayesian Linear Least-Squares:

$$\begin{aligned}&\text{Minimize } E\left[(\underline{\hat{z}} - \underline{z})^T (\underline{\hat{z}} - \underline{z})\right] \\ &\Longrightarrow \underline{\hat{z}}(\underline{m}) = \underline{\mu} + \left(P^{-1} + C^T R^{-1} C\right)^{-1} C^T R^{-1} (\underline{m} - C\underline{\mu})\end{aligned}$$

The algebraic connections among these estimators can be made more explicit by rewriting the latter Bayesian estimator as

$$\begin{aligned}\underline{\hat{z}}(\underline{m}) &= \left\{P^{-1} + C^T R^{-1} C\right\}^{-1} C^T R^{-1} (\underline{m} - C\underline{\mu}) \\ &= \left\{P^{-1} + C^T R^{-1} C\right\}^{-1} \left(C^T R^{-1} \underline{m} + P^{-1} \underline{\mu}\right) \qquad (3.83) \\ &= \underline{\mu} + \left\{ \begin{bmatrix} C^T & I \end{bmatrix} \begin{bmatrix} R^{-1} & 0 \\ 0 & P^{-1} \end{bmatrix} \begin{bmatrix} C \\ I \end{bmatrix} \right\}^{-1} \begin{bmatrix} C^T & I \end{bmatrix} \begin{bmatrix} R^{-1} & 0 \\ 0 & P^{-1} \end{bmatrix} \begin{bmatrix} \underline{m} \\ \underline{\mu} \end{bmatrix}, \qquad (3.84)\end{aligned}$$

in which case, viewed through the context of weighted least-squares (3.26), the Bayesian estimator of (3.84) is equivalent to

$$\begin{bmatrix} \underline{m} \\ \underline{\mu} \end{bmatrix} = \begin{bmatrix} C \\ I \end{bmatrix} \underline{z} + \underline{v}, \qquad \text{minimize } \underline{e}^T \begin{bmatrix} R^{-1} & 0 \\ 0 & P^{-1} \end{bmatrix} \underline{e}. \qquad (3.85)$$

That is, from an algebraic point of view, prior knowledge is equivalent to a measurement: both are pieces of statistical information which guide the estimator. A prior $\underline{z} \sim \mathcal{N}(\underline{\mu}, P)$ may thus be interpreted as saying that we have a "measurement" $\underline{\mu}$ of $\underline{z}$, with the uncertainty in the measurement determined by P.

This duality also leads to a revised, dual criterion for the Bayesian estimator. Our original Bayesian criterion from (3.40),

$$\text{Minimize } E\big[(\hat{\underline{z}} - \underline{z})^T(\hat{\underline{z}} - \underline{z})\big], \tag{3.86}$$

is reinterpreted in the context of weighted least-squares, using the dual formulation (3.84), as the minimization of

$$\left\{\begin{bmatrix}\underline{m}\\ \underline{\mu}\end{bmatrix} - \begin{bmatrix}C\\ I\end{bmatrix}\hat{\underline{z}}\right\}^T \begin{bmatrix}R^{-1} & \mathbf{0}\\ \mathbf{0} & P^{-1}\end{bmatrix}\left\{\begin{bmatrix}\underline{m}\\ \underline{\mu}\end{bmatrix} - \begin{bmatrix}C\\ I\end{bmatrix}\hat{\underline{z}}\right\} \tag{3.87}$$

$$= (\underline{m} - C\hat{\underline{z}})^T R^{-1}(\underline{m} - C\hat{\underline{z}}) + (\underline{\mu} - \hat{\underline{z}})^T P^{-1}(\underline{\mu} - \hat{\underline{z}}) \tag{3.88}$$

$$= \|\underline{m} - C\hat{\underline{z}}\|_{R^{-1}} + \|\underline{\mu} - \hat{\underline{z}}\|_{P^{-1}}\,. \tag{3.89}$$

That is, the Bayesian estimator really is analogous to a standard least-squares approach, except that it simultaneously tries to minimize both the measurement and prior inconsistencies, weighted by their respective confidences R^{-1}, P^{-1}.

3.3 Static Sampling

The notion of *sampling* was introduced in Sections 2.5.2 and 2.5.4, such that we wish to generate random samples of a random vector.

If the prior statistics of a vector are known, we wish to generate prior samples from

$$\underline{z} \sim (\underline{\mu}, P). \tag{3.90}$$

Similarly, if additional constraints beyond the prior model (such as from measurements) are asserted, then random samples may be generated from the constrained vector

$$\underline{z}|\underline{m} \sim (\hat{\underline{z}}, \tilde{P}), \tag{3.91}$$

where $\hat{\underline{z}}, \tilde{P}$ are calculated via the Bayesian estimator (3.65)–(3.68).

As we saw in (2.76), given the mean and covariance of a random vector, random samples can be found by computing a matrix square root (Appendix A.8):

$$\text{Given } \underline{w} \sim I \Rightarrow \text{cov}(\Gamma^T\underline{w}) = E\big[\Gamma^T\underline{w}\,\underline{w}^T\Gamma\big] = \Gamma^T I\Gamma = P. \tag{3.92}$$

Thus

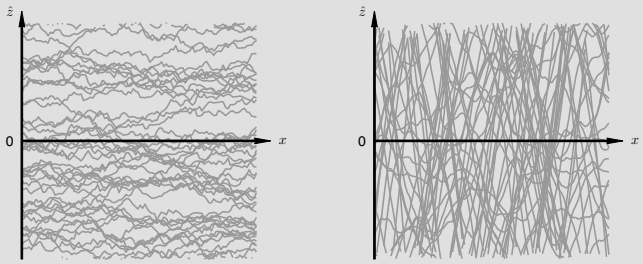
$$\underline{\mu} + P^{1/2}\underline{w} \sim (\underline{\mu}, P) \tag{3.93}$$

$$\hat{\underline{z}} + \tilde{P}^{1/2}\underline{w} \sim (\hat{\underline{z}}, \tilde{P}) \tag{3.94}$$

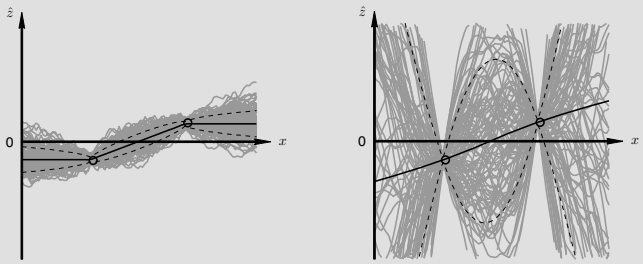
are random prior and posterior samples, respectively. However, compared to computing estimates $\hat{\underline{z}}$, finding prior or posterior samples is *much* more difficult, due to the $\mathcal{O}(n^3)$ calculation of the matrix square root, in addition to the $\mathcal{O}(n^3)$ calculation of the posterior covariance $\tilde{P}$ in the case of posterior sampling.

Example 3.4: Static Estimation and Sampling

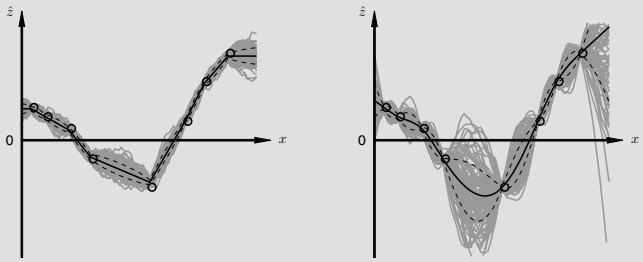
Following up on Example 2.9, suppose we consider an ensemble of random prior samples for a first-order problem (left) or second-order problem (right):



A first-order problem corresponds to an exponential spatial autocorrelation, and the second-order problem very nearly to a Gaussian. Given two measurements, we can clearly observe the posterior samples to be constrained at the measured point, but have greater freedom away from measurements:



Each panel shows the estimate (solid) and one standard-deviation envelope (dashed). It is clear that there is a relationship between the spread of posterior samples and the estimation error variance at any point. Clearly additional measurements more tightly constrain the problem:



Note how the second-order correlation leads to a very smooth interpolation (but also poorly conditioned), whereas the exponential (first-order) case leads to piecewise linear estimates.

An estimation problem which is densely sampled with low-variance measurements leads to $\tilde{P}$ close to zero, in which case the random posterior samples are nearly identical to the estimates $\underline{\hat{z}}$. Conversely, if there are no measurements, then the prior and posterior samples both come from the same distribution, since without measurements

$$\underline{\hat{z}} = \underline{\mu} \qquad \tilde{P} = P. \tag{3.95}$$

If a non-Bayesian static estimation problem is given, as in Section 3.1, then a prior model may not exist if the estimation problem is ill-posed in the absence of measurements. In such cases a formal statistical prior sample does not exist, although in Section 11.2 we show that typical samples can still be generated.

3.4 Data Fusion

From the discussion in Section 2.1 on data fusion, we wanted to allow the possibility of *multiple* measurement sets, rather than just a single measurement vector. Such problems occur frequently in remote sensing, for example, as shown in Figure 3.2, where the same scene is imaged by multiple instruments, possibly at different spatial or temporal resolutions; similar examples occur in computer vision, such as a stereo pair of cameras, and throughout medical imaging, for example combining data from two or more of ultrasound, PET, MRI, and CT images.

In many cases a data fusion problem can be rewritten as a more complicated regular static problem. That is, data fusion problems are not inherently distinct from other estimation or sampling problems; therefore, with the exception of this section, we do not discuss data fusion separately.

Recall from (2.8) in Section 2.1 the basic premise: we are given multiple measurements

$$\underline{m}_1 = C_1\underline{z} + \underline{v}_1 \qquad \underline{v}_1 \sim \mathcal{N}(\underline{0}, R_1) \tag{3.96}$$

$$\underline{m}_2 = C_2\underline{z} + \underline{v}_2 \qquad \underline{v}_2 \sim \mathcal{N}(\underline{0}, R_2) \tag{3.97}$$

$$\vdots \qquad\qquad \vdots$$

It is possible to stack these multiple measurement models into a single one:

$$\underline{m} = \begin{bmatrix} \underline{m}_1 \\ \underline{m}_2 \\ \vdots \end{bmatrix} = \begin{bmatrix} C_1 \\ C_2 \\ \vdots \end{bmatrix} \underline{z} + \begin{bmatrix} \underline{v}_1 \\ \underline{v}_2 \\ \vdots \end{bmatrix} \quad \underline{v} = \begin{bmatrix} \underline{v}_1 \\ \underline{v}_2 \\ \vdots \end{bmatrix} \sim \mathcal{N}\left(\underline{0}, \begin{bmatrix} R_1 & \mathbf{0} & \cdots \\ \mathbf{0} & R_2 & \mathbf{0} \\ \vdots & \mathbf{0} & \ddots \end{bmatrix} \right) \tag{3.98}$$

which is a basic static problem, where the block-diagonal nature of the measurement noise follows from the assumption that the various measurement noise terms $\underline{v}_1, \underline{v}_2, \ldots$ are uncorrelated. It is important to note, however, that the conversion back

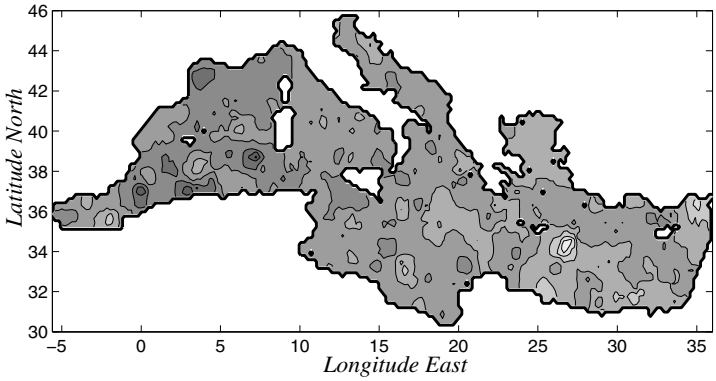


Fig. 3.2. An example of Bayesian estimation and data fusion [109]: A static Bayesian estimator (based on Section 10.3) produced these estimates of sea-surface height in the Mediterranean Sea. The estimates are based on fusing the altimetric (height) data from two complementary satellites: Topex–Poseidon [120], which samples more sparsely in space but more frequently in time, and ERS-1 [203], which samples more densely in space but less frequently in time.

to the static form is *not* dependent on noise decorrelation, or even on the linearity of the measurement model. Given nonlinear measurements having correlated noise,

$$\underline{m}_1 = C_1(\underline{z}) + \underline{v}_1 \qquad \underline{v}_1 \sim \mathcal{N}(\underline{0}, R_{11}) \tag{3.99}$$

$$\underline{m}_2 = C_2(\underline{z}) + \underline{v}_2 \qquad \underline{v}_2 \sim \mathcal{N}(\underline{0}, R_{22}) \tag{3.100}$$

$$\vdots \qquad\qquad \vdots$$

the problem transforms as in (3.98):

$$\underline{m} = \begin{bmatrix} \underline{m}_1 \\ \underline{m}_2 \\ \vdots \end{bmatrix} = \begin{bmatrix} C_1(\underline{z}) \\ C_2(\underline{z}) \\ \vdots \end{bmatrix} + \begin{bmatrix} \underline{v}_1 \\ \underline{v}_2 \\ \vdots \end{bmatrix} \quad \underline{v} = \begin{bmatrix} \underline{v}_1 \\ \underline{v}_2 \\ \vdots \end{bmatrix} \sim \mathcal{N}\left(\underline{0}, \begin{bmatrix} R_{11} & R_{12} & \cdots \\ R_{21} & R_{22} & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} \right). \tag{3.101}$$

A more interesting case is one in which the measurements arrive separately. Suppose we first receive a set of measurements, from which we compute estimates:

$$\left.\begin{array}{ll} \text{Initial prior} & \underline{z} \sim (\underline{0}, P) \\ \text{First measurement} & \underline{m}_1 = C_1\underline{z} + \underline{v}_1 \end{array}\right\} \quad \underline{z}|\underline{m}_1 \sim \left(\hat{\underline{z}}_1, \tilde{P}_1\right). \tag{3.102}$$

Then, suppose that afterward a second set of measurements becomes available. We know that we could combine the first and second set together, as in (3.98), however that requires that we hold on to the previous measurements $\underline{m}_1$ indefinitely, and also

it throws away the work that we did in (3.102) to compute the estimates. Really, we want to *fuse* the new measurements into the existing estimates. Actually this is not so hard, because we can view the *posterior* $(\underline{\hat{z}}_1, \tilde{P}_1)$ from the first stage as the *prior* to the second:

$$\left.\begin{array}{ll} \text{Current "prior"} & \underline{z}|\underline{m}_1 \sim (\underline{\hat{z}}_1, \tilde{P}_1) \\ \text{Second measurement} & \underline{m}_2 = C_2\underline{z} + \underline{v}_2 \end{array}\right\} \quad \underline{z}|\underline{m}_1, \underline{m}_2 \sim \left(\underline{\hat{z}}_{12}, \tilde{P}_{12}\right). \quad (3.103)$$

This latter step is just another, regular static problem, where we can see that the calculation of the error statistics $\tilde{P}_1$ in the former step was essential to fuse the later measurements.

The above, sequential process in (3.102), (3.103) is something like a dynamic problem, in that measurements are arriving one after another, however the underlying state $\underline{z}$ is not changing, so there are actually no dynamics present. Nevertheless, this example has much in common with dynamic estimation, and the above idea leads particularly nicely into the dynamic Kalman filter, derived in Chapter 4.

Application 3: Atmospheric Temperature Inversion [282]

There are a number of scientific challenges which follow the generic inverse problem very closely:

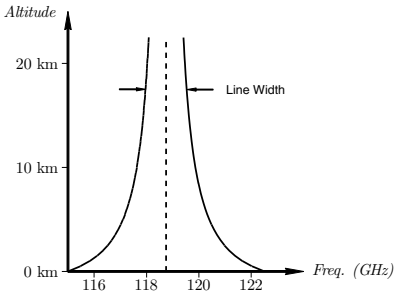
$$\text{Underlying State} \xrightarrow{\text{Forward}} \text{Measurements} \xrightarrow{\text{Inverse}} \text{Estimated State}$$

One such example is the temperature inverse-problem of the atmosphere [282]. Although temperatures can be measured directly at one location using a weather balloon, to produce a map of temperature with large-scale coverage it would be preferable to use a satellite or high-flying aircraft to *infer* the temperature remotely.

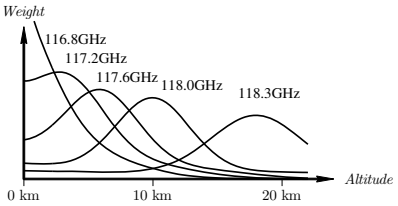
UNDERLYING STATE:
We wish to estimate the atmospheric temperature $t(a)$ as a function of altitude[2] a.

FORWARD PROBLEM:
It is well known that every molecule (such as oxygen O_2) is associated with a line (emission / absorption) spectrum. The spectral line is not infinitesimally thin, however, rather it is a smeared line whose width is a function of pressure (the *Stark effect*), as sketched in Figure 3.3(a). Furthermore, the strength with which Oxygen radiates energy is a function of its temperature. Therefore a satellite, far above the atmosphere, observing the microwave energy at 116 GHz will be able to infer

[2] Technically the temperature is normally expressed as a function of *pressure*, however ignoring spatial variations in atmospheric pressure, there is a one-to-one relationship between pressure and altitude.



(a)
Spectral Line Width vs. Altitude



(b)
Altitude Weighting Functions
for Different Frequencies

Fig. 3.3. The forward problem for atmospheric temperature inversion. The width of the oxygen 118 GHz spectral line, left, varies with pressure (or, equivalently, altitude). By solving the process by which microwave radiation moves through the atmosphere, right, we obtain a set of curves, each describing how much a given frequency is influenced by temperature at a given altitude.

ground-level temperature, since at all altitudes above the ground the atmosphere is transparent to 116 GHz.

The problem is a bit more complicated, however. The spectral power around 118 GHz, closer to the spectral line, is *not* just the average of atmospheric temperature from 0 km to 15 km. Instead, a microwave photon emitted near the ground may be absorbed, and re-emitted, by another molecule higher in the atmosphere (the so-called *radiative transfer* problem). If this process is simulated, we can find the weighting functions $w(a, f)$ at altitude a and frequency f, such that the spectral strength observed from space is

$$s(f) = \int t(a) \cdot w(a, f)\, da. \tag{3.104}$$

Normally our satellite would observe the spectrum at only a discrete number of frequencies, as suggested by the sketch in Figure 3.3(b).

INVERSE PROBLEM:

We suppose that we wish to estimate temperatures at discrete altitudes $a_1, \ldots, a_n$. Furthermore we suppose that the satellite observes the spectrum at frequencies $f_1, \ldots, f_q$. Our state and measurement are therefore defined,

$$\underline{z} = \begin{bmatrix} t(a_1) \\ \vdots \\ t(a_n) \end{bmatrix} \qquad \underline{m} = \begin{bmatrix} s(f_1) \\ \vdots \\ s(f_q) \end{bmatrix} \tag{3.105}$$

leading to a forward problem

$$\underline{m} = W\underline{z} + \underline{v}. \tag{3.106}$$

W follows from the weights in Figure 3.3(b) and $\underline{v}$ is a noise term which will be a function of the instrument design. Normally this problem will be underconstrained, so we want to regularize $\underline{z}$, most likely by asserting smoothness.

Because this inverse problem is one-dimensional (a single look down through the atmosphere), the number of unknowns is modest and the problem can be solved directly by matrix inversion. Analytical methods and neural networks have also been used.

Summary

Given a least-squares problem $\underline{m} = C\underline{z}$, our objective is to minimize the weighted squared error

$$\underline{\hat{z}} = \arg_{\underline{z}} \min \left\{ (\underline{m} - C\underline{z})^T W (\underline{m} - C\underline{z}) \right\} \tag{3.107}$$

for which the optimum solution is given by

$$\underline{\hat{z}} = \left(C^T W C \right)^{-1} C^T W \underline{m}. \tag{3.108}$$

Given a constrained, regularized problem our objective is to minimize the weighted squared error

$$\min \left\{ \|\underline{m} - C\underline{\hat{z}}\|_{R^{-1}} + \lambda \|L\underline{\hat{z}}\| \right\} \tag{3.109}$$

for which the optimum solution is given by

$$\underline{\hat{z}} = \left(C^T R^{-1} C + \lambda L^T L \right)^{-1} C^T R^{-1} \underline{m}. \tag{3.110}$$

Given a static Bayesian estimation problem

$$\underline{z} \sim (\underline{\mu}, P) \qquad \underline{m} = C\underline{z} + \underline{v} \qquad \underline{v} \sim (\underline{0}, R), \tag{3.111}$$

the optimum estimator can be formulated in the following two algebraically-equivalent ways:

Form I:

$$\underline{\hat{z}}(\underline{m}) = \underline{\mu} + (PC^T)(CPC^T + R)^{-1}(\underline{m} - C\underline{\mu}) \tag{3.112}$$

$$\tilde{P} = \text{cov}(\underline{\tilde{z}}) = P - (PC^T)(CPC^T + R)^{-1}CP \tag{3.113}$$

Form II:

$$\underline{\hat{z}}(\underline{m}) = \underline{\mu} + \left(P^{-1} + C^T R^{-1} C \right)^{-1} C^T R^{-1} (\underline{m} - C\underline{\mu}) \tag{3.114}$$

$$\tilde{P} = \text{cov}(\underline{\tilde{z}}) = \left(P^{-1} + C^T R^{-1} C \right)^{-1} \tag{3.115}$$

For Further Study

There are a great many textbooks that cover estimation theory; as an accessible introduction, the reader is referred to [248, 284]. For a comprehensive coverage of spatial statistics, the book by Cressie [75] is an excellent reference.

Sample Problems

Problem 3.1: Static Estimation Forms

Use the ABCD identity from Appendix A to prove the equivalence of the two static estimator forms (3.112)–(3.113) and (3.114)–(3.115).

Problem 3.2: Linear Regression

In Example 3.1 on page 62, the answer to the estimate $\hat{\underline{z}}$ is written down, but without derivation. Complete the example by deriving (3.20), the solution to $\hat{a}, \hat{b}$.

Problem 3.3: Static Sampling

Suppose you are given a vector $\underline{\mu}$ and a covariance P. Prove that

$$\left(\underline{\mu} + P^{1/2}\underline{w}\right) \quad \text{is distributed as} \quad (\underline{\mu}, P),$$

where $\underline{w} \sim I$.

Problem 3.4: Covariance Reduction

A measurement should never *increase* our uncertainty in an estimate. At worst a measurement is useless and should have no effect, in all other cases the estimation error covariance should decrease from the prior. Prove that this is so.

That is, for Bayesian least squares prove that $\tilde{P} \leq P$, where the matrix inequality is understood in positive-semidefinite terms; in other words, that $P - \tilde{P}$ must be positive-semidefinite. The proof is easiest beginning with Form I (3.113).

Problem 3.5: First-Order Interpolation

Let $\underline{z}$ be a vector having 100 elements. We wish to do regularized, constrained estimation as in (3.109), (3.110).

We assert a first-order (gradient) constraint in L, penalizing the differences between adjacent state elements. Thus L is a 99×100 matrix such that

$$l_{i,j} = \begin{cases} \text{-1} & \text{if } j = i \\ 1 & \text{if } j = i+1 \\ 0 & \text{otherwise} \end{cases}$$

meaning that each row of L looks like

$$\begin{bmatrix} 0 \ldots 0 & \text{-1} & 1 & 0 & 0 \ldots \end{bmatrix}.$$

(a) How many measurements are required for the constrained problem to satisfy uniqueness? Why?
(b) Suppose we observe elements z_{25} and z_{75}, thus

$$\underline{m} = C\underline{z} + \underline{v} = \begin{bmatrix} z_{25} \\ z_{75} \end{bmatrix} + \underline{v} \quad \text{where} \quad \underline{m} = \begin{bmatrix} 10 \\ 50 \end{bmatrix} \quad \underline{v} \sim I.$$

Compute and plot $\hat{\underline{z}}$ for $\lambda = 1, 5, 25$.

Problem 3.6: Second-Order Interpolation

Let $\underline{z}$ be a vector having 100 elements. We wish to do regularized, constrained estimation as in (3.109), (3.110).

Following on Problem 3.5, we assert a second-order (curvature) constraint in L, penalizing the curvature (rate of change of gradient) across three successive state elements. Thus L is a 98×100 matrix such that

$$l_{i,j} = \begin{cases} \text{-1} & \text{if } j = i, i+2 \\ 2 & \text{if } j = i+1 \\ 0 & \text{otherwise} \end{cases}$$

so that each row of L looks like

$$\begin{bmatrix} 0 \ldots 0 & \text{-1} & 2 & \text{-1} & 0 & 0 \ldots \end{bmatrix}.$$

(a) How many measurements are required for the constrained problem to satisfy uniqueness? Why?
(b) Suppose we observe elements z_1, z_{50}, and z_{100} with

$$\underline{m}^T = \begin{bmatrix} 0 & 50 & 25 \end{bmatrix} \quad \underline{v} \sim I.$$

Compute and plot $\hat{\underline{z}}$ for $\lambda = 250, 2000, 10000$.

(c) Comment on the differences in the estimates produced by first- and second-order constraints.

Problem 3.7: Cross-Validation

We wish to use the method of cross-validation (2.56) on Page 37 from Chapter 2 to infer the optimal regularization constraint λ in Problems 3.5 and 3.6. A part of this problem was solved in Example 2.8; the reader may wish to refer back to this example.

Suppose we have three measurement sets:

$$\begin{aligned} m_i^{(1)} &= 5 + v_i^{(1)} \\ m_i^{(2)} &= \tfrac{i}{10} + v_i^{(2)} \\ m_i^{(3)} &= \left(\tfrac{i-50}{20}\right)^2 + v_i^{(3)} \end{aligned}$$

where $1 \leq i \leq 100$ and where $v_i^{(j)}$ is a white noise process of variance 2.0.

For each of the measurement sets, and for each of the first- and second-order constraints, perform cross-validation.

Plot the measurement–estimate inconsistency $e^2(\lambda)$ from (2.56), find the optimal value of λ, and plot the estimates corresponding to $\lambda/100$, λ, and 100λ.

Comment on the behaviour of the $e^2(\lambda)$ and estimation plots.

Problem 3.8: Data Fusion

Let's investigate doing data fusion. Suppose $\underline{z}$ is a vector of 100 unknowns, where we observe elements $z_{10}, z_{30}, z_{50}, z_{70}$, and z_{90} with

$$\underline{m}^T = \begin{bmatrix} 0 & 0 & 20 & 30 & 0 \end{bmatrix} \quad \underline{v} \sim I.$$

The data fusion problem needs a prior; we use the second-order constraint of Problem 3.6, such that the prior $P^{-1} = \lambda L^T L$, where $\lambda = 2000$. We'll compute estimates in one of two ways:

(a) Using usual constrained least squares, compute the estimates $\underline{\hat{z}}(\underline{m})$ and estimation error covariance $\tilde{P}$. Plot the five measurements, superimposed on the estimates, plus–minus one standard deviation

$$\underline{\hat{z}} \pm \sqrt{\text{diag}\left(\tilde{P}\right)}.$$

(b) Now we want to do data fusion, whereby we will incorporate only one measurement at a time, using the method outlined in (3.102), (3.103):

1. Compute $\underline{\hat{z}}_2, \tilde{P}_2$ from measurements m_1, m_2, using prior P as before.

2. Compute $\underline{\hat{z}}_3, \tilde{P}_3$ by fusing measurement m_3 into $\underline{\hat{z}}_2, \tilde{P}_2$.
3. Compute $\underline{\hat{z}}_4, \tilde{P}_4$ by fusing m_4 into $\underline{\hat{z}}_3, \tilde{P}_3$.
4. Compute $\underline{\hat{z}}_5, \tilde{P}_5$ by fusing m_5 into $\underline{\hat{z}}_4, \tilde{P}_4$.

Plot

$$\underline{\hat{z}}_i \pm \sqrt{\text{diag}\big(\tilde{P}_i\big)},$$

superimposed on the relevant measurement, for each iteration $i = 2, \ldots, 5$.

State your observations, comparing the results from (a) and (b).

4

Dynamic Estimation and Sampling

Chapter 3 developed estimators for static problems — those in which $\underline{z}$ has no time dependence. Given measurements, we compute a corresponding set of estimates, and the solution is complete.

Clearly the problem is much more interesting if $\underline{z}$ is permitted to evolve and be measured over time. Indeed, this is a classic problem in the control literature, in which we wish to control some aspects (such as temperature, velocity, pressure, height, volume, etc.) of a time-evolving system $\underline{z}$. A common step in developing a controller is to develop an "observer," to estimate $\hat{\underline{z}}$, the unknown current state of the system. The Kalman filter [182] is a dynamic, recursive estimator which was first developed as an observer for system control, but which has found fantastically broad application in a wide variety of fields, including statistical image processing.

Although temporal image processing is certainly an area of interest, particularly for video sequences, it is important to realize that the "time" t may be interpreted much more abstractly: *any* signal which varies as a function of a discrete variable is a candidate for Kalman filtering. Thus, in the context of multidimensional signal processing, we may choose t to index over time, over video frames, over the rows or columns of an image, over the planes in a 3D volume, over the resolutions in a multiresolution tree, or the raster scan of a 2D image, to name only a few possibilities. Three real, practical examples are illustrated in Figure 4.1.

To be sure, derivations of the Kalman filter may be found in many texts [8, 97, 151, 156, 231, 284]. We repeat one here, partly for reasons of continuity in development, but also because many other derivations are unnecessarily complicated, and may include continuous-time derivations, which are important in control, but have little to contribute to the inherently discrete world of pixellated images.

Our discussion begins with a consideration of first-order Gauss–Markov models, and of the relationship between static and dynamic estimation problems, followed by a derivation of the discrete-time Kalman filter.

P. Fieguth, *Statistical Image Processing and Multidimensional Modeling*, Information Science and Statistics, DOI 10.1007/978-1-4419-7294-1_4, © Springer Science+Business Media, LLC 2011

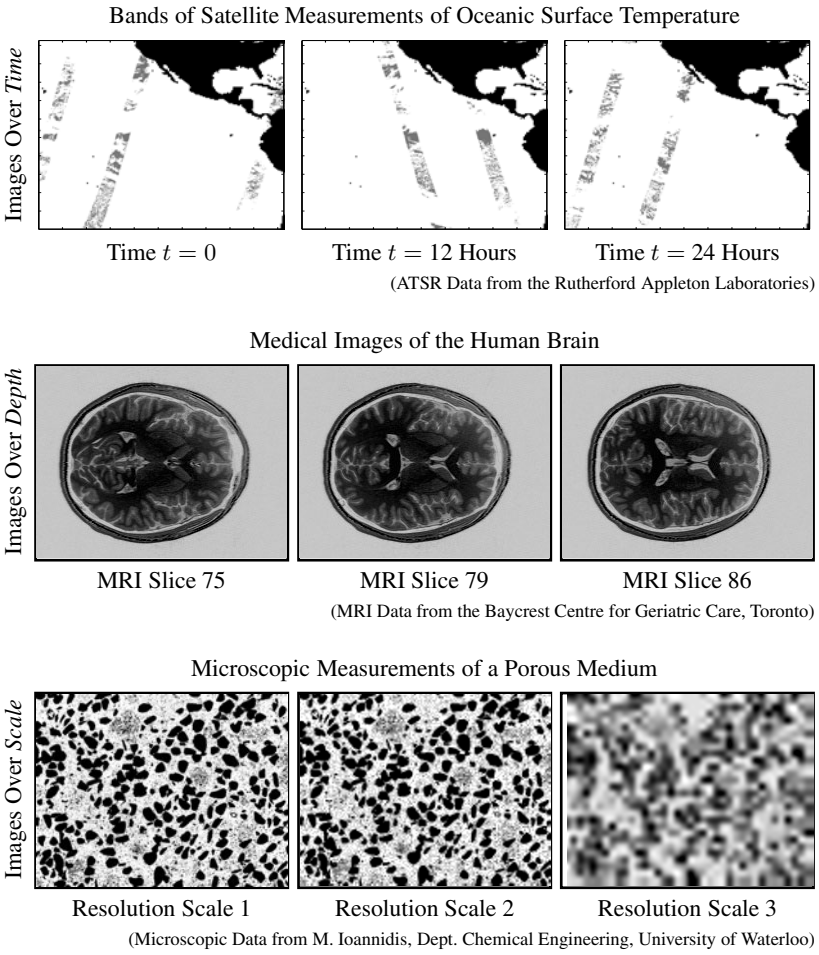


Fig. 4.1. There are many circumstances in which we have image sequences. Whether the sequence is indexed over time (top), over space (middle), or over resolution (bottom), all of these can be modelled as dynamic processes.

4.1 The Dynamic Problem

We encountered a basic, canonical dynamic problem in Section 2.5.1. Our starting point here is the most famous of all linear systems, the first-order Gauss–Markov dynamic model [8, 248]

$$\underline{z}(t+1) = A(t)\underline{z}(t) + B(t)\underline{w}(t) \qquad \underline{w}(t) \sim \mathcal{N}(\underline{0}, I), \tag{4.1}$$

in which the random process $\underline{w}$ is white and uncorrelated with the past of the system:

$$E\left[\underline{w}(t)\underline{w}(s)^T\right] = \delta_{t,s}I, \qquad E\left[\underline{w}(t)\underline{z}(s)^T\right] = \mathbf{0} \text{ if } s \leq t. \tag{4.2}$$

The interpretation or purpose of the *process noise* $\underline{w}$ can be context dependent:

- In some cases our dynamic system $\underline{z}(t)$ is *actually* stochastic, and really *is* driven by a white noise process, such that $\underline{w}$ is adding energy into the system. This is true for a variety of physical phenomena, the most famous of which is the random walk of Brownian motion.
- Our mathematical *model* for $\underline{z}(t)$ may be deterministic, however we may still choose to include a noise term $\underline{w}(t)$. A deterministic estimator will become increasingly confident over time, to the point where it will refuse to adjust the estimates to changing measurements. A small amount of modelled noise limits the estimator confidence and preserves adaptability.
- The mathematical model $A(t), B(t)$ may, in many cases, be only an approximation of the real world. For example, in developing a model for the motion of a car, details such as variations in tire pressure, small bumps on the road, wind, and turbulence may not be modelled. Although more formal methods exist for analyzing and dealing with model errors, a noise term in the dynamics gives the estimator a bit of flexibility in accommodating model approximation.

Next, a boundary condition is required to initiate the recursion:

$$E[\underline{z}(0)] = \underline{z}_o \qquad \text{cov}(\underline{z}_o) = P_o. \tag{4.3}$$

Finally, measurements are available over time:

$$\underline{m}(t) = C(t)\underline{z}(t) + \underline{v}(t) \qquad \underline{v}(t) \sim \mathcal{N}\big(\underline{0}, R(t)\big). \tag{4.4}$$

Taken together, the dynamic equation, boundary condition, and measurements form a complete description of the canonical dynamic problem from Section 2.5.1. Let

$\hat{\underline{z}}(t|s)$ be the least-squares estimate of $\underline{z}(t)$ given $\underline{m}(0), \ldots, \underline{m}(s)$, and
$P(t|s)$ be the estimation error covariance corresponding to $\hat{\underline{z}}(t|s)$.

Then, given $\underline{m}(0), \ldots, \underline{m}(\tau)$ we have two fundamental problems:

Filtering: find $\hat{\underline{z}}(t|t)$ for $0 \leq t \leq \tau$
Smoothing: find $\hat{\underline{z}}(t|\tau)$ for $0 \leq t \leq \tau$

4.1.1 First-Order Gauss–Markov Processes

The proposed model (4.1) is a first-order, Gauss–Markov, discrete-time random process. At first glance it is unclear to what extent this model has a sufficiently general form. For example, two simple, common, alternative models which do not immediately fit into the form of (4.1) are an nth-order autoregressive model [37]

$$\underline{z}(t+1) = \sum_{i=0}^{n-1} A_i(t)\underline{z}(t-i) + B(t)\underline{w}(t) \tag{4.5}$$

or an nth-order correlated-noise (moving-average) model

$$\underline{z}(t+1) = \sum_{i=0}^{n-1} B_i(t)\underline{w}(t-i). \tag{4.6}$$

In fact, both of these forms *can* be rewritten into the form of (4.1), thus any estimator compatible with (4.1) does in fact generalize to a much broader range of models. We illustrate this generalization for two examples. Given an autoregressive model of the form (4.5)

$$\underline{z}(t+1) = \sum_{i=0}^{2} A_i(t)\underline{z}(t-i) + B(t)\underline{w}(t), \tag{4.7}$$

then under the following definition of variables

$$\bar{\underline{z}}(t) \triangleq \begin{bmatrix} \underline{z}(t) \\ \underline{z}(t-1) \\ \underline{z}(t-2) \end{bmatrix} \qquad \bar{A}(t) \triangleq \begin{bmatrix} A_0(t) & A_1(t) & A_2(t) \\ I & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & I & \mathbf{0} \end{bmatrix} \qquad \bar{B}(t) \triangleq \begin{bmatrix} B(t) \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} \tag{4.8}$$

we obtain a first-order dynamic recursion, having the canonical form of (4.1):

$$\bar{\underline{z}}(t+1) = \bar{A}(t)\bar{\underline{z}}(t) + \bar{B}(t)\underline{w}(t). \tag{4.9}$$

A second example illustrates the same idea for the moving-average process of (4.6). Given the process

$$\underline{z}(t+1) = \sum_{i=0}^{3} B_i(t)\underline{w}(t-i), \tag{4.10}$$

we can formulate a change of variables

$$\bar{\underline{z}}(t) \triangleq \begin{bmatrix} \underline{z}(t) \\ \underline{w}(t-1) \\ \underline{w}(t-2) \\ \underline{w}(t-3) \end{bmatrix} \qquad \bar{A}(t) \triangleq \begin{bmatrix} \mathbf{0} & B_1(t) & B_2(t) & B_3(t) \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & I & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & I & \mathbf{0} \end{bmatrix} \qquad \bar{B}(t) \triangleq \begin{bmatrix} B_0(t) \\ I \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} \tag{4.11}$$

such that the following Gauss–Markov model implements the moving average of (4.10):

$$\underline{\bar{z}}(t+1) = \bar{A}(t)\underline{\bar{z}}(t) + \bar{B}(t)\underline{w}(t). \tag{4.12}$$

Thus we claim that, indeed, the proposed first-order Gauss–Markov model (4.1) can encompass a wide variety of models, both first- and higher-order.

Next, it should be clarified that the *entire* prior model over all time $t \geq 0$ is specified, implicitly, by the boundary condition at time $t = 0$. Given

$$E\big[\underline{z}(0)\big] = \underline{z}_o \qquad \text{cov}(\underline{z}_o) = P_o, \tag{4.13}$$

we can find a recursive form for the mean $E\big[\underline{z}(t)\big]$,

$$\begin{aligned} E\big[\underline{z}(t+1)\big] &= E\big[A(t)\underline{z}(t) + B(t)\underline{w}(t)\big] && (4.14)\\ &= A(t)E\big[\underline{z}(t)\big] && (4.15)\\ &= A(t)A(t-1)\ldots A(0)\underline{z}_o && (4.16) \end{aligned}$$

and similarly for the covariance $P(t)$,

$$\begin{aligned} P(t+1) &= \text{cov}\big(A(t)\underline{z}(t) + B(t)\underline{w}(t)\big) && (4.17)\\ &= A(t)P(t)A^T(t) + B(t)B^T(t). && (4.18) \end{aligned}$$

If the system dynamics are time-invariant, $A(t) = A$, and stable, meaning that all of the eigenvalues of A have a magnitude less than one, then the process converges to a statistical steady-state. That is, $\underline{z}(t)$ continues to change and evolve, but *all* of the statistics of $\underline{z}(t)$ converge. In particular, the mean converges to zero,

$$\lim_{t\to\infty} E\big[\underline{z}(t)\big] = \lim_{t\to\infty} A^t\underline{z}_o = \underline{0}, \tag{4.19}$$

and the process covariance converges to the solution of the Algebraic Lyapunov Equation:

$$\lim_{t\to\infty} P(t) = P_\infty \qquad \text{where} \qquad P_\infty = AP_\infty A^T + BB^T. \tag{4.20}$$

The steady-state case is of some interest in Kalman filtering and will be revisited in Section 4.3.2.

4.1.2 Static — Dynamic Duality

There is nothing inherently difficult in estimating a dynamic quantity. In fact, it is possible to convert the dynamic problem into a static one, and then to use the methods of Chapter 3 which we have derived for static estimation.

Consider the problem of estimating $\underline{\hat{z}}(t|t)$. Suppose we create augmented state and measurement vectors

$$\bar{\underline{z}}(t) = \begin{bmatrix} \underline{z}(0) \\ \vdots \\ \underline{z}(t) \end{bmatrix} \qquad \bar{\underline{m}}(t) = \begin{bmatrix} \underline{m}(0) \\ \vdots \\ \underline{m}(t) \end{bmatrix} \tag{4.21}$$

Then solving the estimation problem produces estimates

$$\hat{\bar{\underline{z}}}(t) = \begin{bmatrix} \hat{\underline{z}}(0|t) \\ \vdots \\ \hat{\underline{z}}(t|t) \end{bmatrix} \tag{4.22}$$

of each element of $\bar{\underline{z}}$ given all of the measurements in $\bar{\underline{m}}$, where the estimates include the desired $\hat{\underline{z}}(t|t)$.

If we concatenate the dynamic measurement model as

$$\bar{\underline{v}}(t) = \begin{bmatrix} \underline{v}(0) \\ \vdots \\ \underline{v}(t) \end{bmatrix} \qquad \bar{C}(t) = \begin{bmatrix} C(0) & & \\ & \ddots & \\ & & C(t) \end{bmatrix} \qquad \bar{R}(t) = \begin{bmatrix} R(0) & & \\ & \ddots & \\ & & R(t) \end{bmatrix} \tag{4.23}$$

where $\bar{C}(t), \bar{R}(t)$ are block-diagonal, then we have recovered the measurement model corresponding to the following static problem:

$$\bar{\underline{m}} = \bar{C}\bar{\underline{z}} + \bar{\underline{v}} \qquad \bar{\underline{z}} \sim (\bar{\underline{\mu}}, \bar{P}) \qquad \bar{\underline{v}} \sim (\underline{0}, \bar{R}). \tag{4.24}$$

What remains is the more difficult rearrangement of the dynamic model for $\underline{z}(t)$ to infer the prior covariance $\bar{P}$ of the above static problem. We begin with the straightforward computation of the prior mean; from (4.16) it follows that

$$\bar{\underline{\mu}}(t) = E\big[\bar{\underline{z}}(t)\big] = E\begin{bmatrix} \underline{z}(0) \\ \underline{z}(1) \\ \vdots \\ \underline{z}(t) \end{bmatrix} = \begin{bmatrix} \underline{z}_o \\ A(0)\underline{z}_o \\ \vdots \\ A(t-1)\cdot\ldots\cdot A(0)\underline{z}_o \end{bmatrix} \tag{4.25}$$

Next, to compute the prior covariance $\bar{P}$, observe that

$$\begin{aligned} &E\Big[\big(\underline{z}(t+1) - \underline{\mu}(t+1)\big)\big(\underline{z}(t) - \underline{\mu}(t)\big)^T\Big] \\ &\quad = E\Big[\big(A(t)\underline{z}(t) + B(t)\underline{w}(t) - A(t)\underline{\mu}(t)\big)\big(\underline{z}(t) - \underline{\mu}(t)\big)^T\Big] \\ &\quad = A(t)E\Big[\big(\underline{z}(t) - \underline{\mu}(t)\big)\big(\underline{z}(t) - \underline{\mu}(t)\big)^T\Big] + B(t)E\Big[\underline{w}(t)\big(\underline{z}(t) - \underline{\mu}(t)\big)^T\Big] \\ &\quad = A(t)P(t) + \mathbf{0}. \end{aligned} \tag{4.26}$$

Iterating (4.26), we can find all needed cross-correlations

$$E\Big[\big(\underline{z}(t+s)-\underline{\mu}(t+s)\big)\big(\underline{z}(t)-\underline{\mu}(t)\big)^T\Big] = A(t+s-1)\cdot\ldots\cdot A(t)P(t)$$
$$E\Big[\big(\underline{z}(t)-\underline{\mu}(t)\big)\big(\underline{z}(t+s)-\underline{\mu}(t+s)\big)^T\Big] = P(t)A^T(t)\cdot\ldots\cdot A^T(t+s-1) \tag{4.27}$$

for $s>0$, from which follows the massive covariance $\bar{P}(t)=\mathrm{cov}\big(\underline{\bar{z}}(t)\big)$:

$$\bar{P}(t)=\begin{bmatrix} P(0) & P(0)A^T(0) & P(0)A^T(0)A^T(1)\cdots & \\ A(0)P(0) & P(1) & P(1)A^T(1) & \cdots \\ A(1)A(0)P(0) & A(1)P(1) & P(2) & \\ \vdots & \vdots & \vdots & \ddots \\ & & & P(t) \end{bmatrix} \tag{4.28}$$

With the definition of the static problem complete, we can use the results of Section 3.2.1 to find the estimates:

$$\hat{\underline{\bar{z}}}=\begin{bmatrix}\hat{\underline{z}}(0|t)\\ \vdots \\ \hat{\underline{z}}(t|t)\end{bmatrix}=\bar{\underline{\mu}}+\big(\bar{C}^T\bar{R}^{-1}\bar{C}+\bar{P}^{-1}\big)^{-1}\bar{C}^T\bar{R}^{-1}(\bar{\underline{m}}-\bar{C}\bar{\underline{\mu}}), \tag{4.29}$$

from which the desired estimate, $\hat{\underline{z}}(t|t)$, can be extracted as the last element of $\hat{\underline{\bar{z}}}$.

The problem with this dual approach, which becomes obvious when looking at (4.28), is that the size of the vectors and matrices *grows* with increasing t. That is, the problem becomes increasingly difficult over time. For example, if the system state is n-dimensional, $\underline{z}(t)\in\mathbb{R}^n$, then the computation of $\hat{\underline{z}}(t|t)$ involves vectors of length $(t+1)n$ and a matrix-inversion complexity of $\mathcal{O}\big((tn)^3\big)$.

There is also something dissatisfying about this dual approach, in that there is a great deal of duplicated effort in computing $\hat{\underline{z}}(t|t)$ and $\hat{\underline{z}}(t+1|t+1)$. In particular, observe that the problem formulation at time $t+1$ shares a great deal in common with that at time t:

$$\bar{C}(t+1)=\begin{bmatrix}\bar{C}(t) & \mathbf{0}\\ \mathbf{0} & C(t+1)\end{bmatrix}\qquad \bar{R}(t+1)=\begin{bmatrix}\bar{R}(t) & \mathbf{0}\\ \mathbf{0} & R(t+1)\end{bmatrix} \tag{4.30}$$

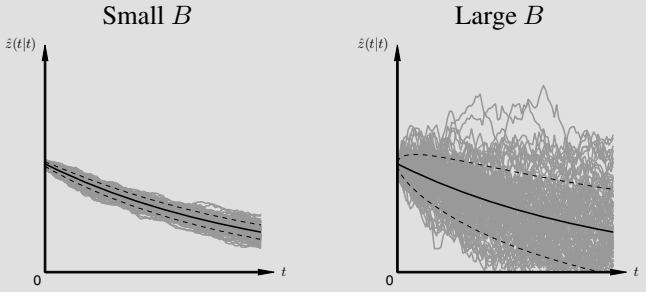
$$\bar{P}(t+1)=\begin{bmatrix}\bar{P}(t) & D\\ D^T & P(t+1)\end{bmatrix} \tag{4.31}$$

The key to finding a more efficient estimator is to look again at the original recursive dynamics in (4.1). In particular, note that the state of the system at time $t+1$ depends *only* on the state at time t; that is, the state $\underline{z}(t)$ somehow captures or summarizes the entire past history of the process[1] that is relevant to $\underline{z}(t+1)$. The derivation in Section 4.2 shows that such a recursive form *also* exists for the estimator $\hat{\underline{z}}(t)$.

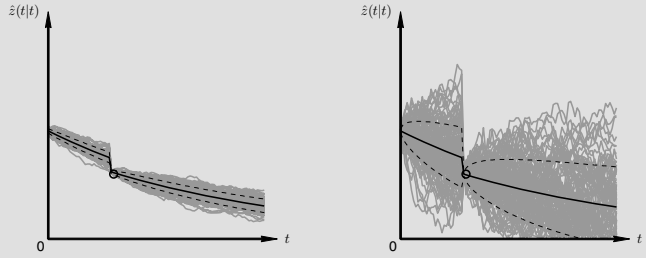
[1] Which is essentially a statement of Markovianity, a subject which is discussed further in Chapter 6.

Example 4.1: Dynamic Estimation and Sampling

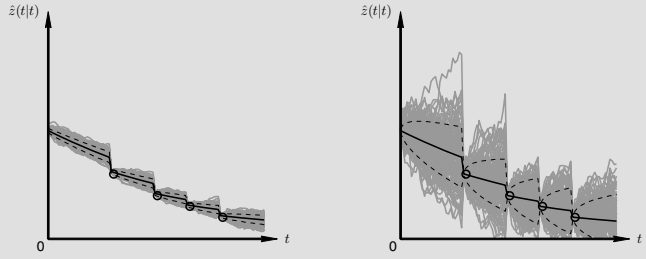
Following up on Examples 2.9 and 3.4, suppose we have two random processes, where B in (4.1) controls the amount of process noise:



Both processes are scalar, first-order, with $A \approx 0.99$. The solid line plots the estimates, and the dashed lines the one standard-deviation envelope. Clearly the rate at which the uncertainty grows is a function of B. Now suppose a measurement is introduced:



Because the estimates are causal, $\hat{z}(t|t)$ is unaware of any measurement until the measurement occurs. Each measurement constrains the posterior samples of the random process, with multiple measurements introducing multiple constraints:



Solving for these dynamic estimates is not hard, and follows from (4.29). What is unique about the Kalman filter, derived in Section 4.2, is an *efficient, recursive* approach to producing dynamic estimates such as these.

4.2 Kalman Filter Derivation

We seek to discover a recursive form for the estimator $\underline{z}(t|t)$. It is common to encounter Kalman filter derivations in textbooks which *assert* the recursive form,

$$\hat{\underline{z}}(t+1|t+1) = H(t)\hat{\underline{z}}(t|t) + K(t)\underline{m}(t) + \underline{\alpha}(t), \tag{4.32}$$

and then derive the optimum settings for $H(t), K(t), \underline{\alpha}(t)$. However, such an approach does two disservices to the reader:

1. By asserting the recursive form *ab initio*, one finds only the best recursive form, without necessarily showing that the recursive form is, in fact, the best overall estimator.
2. By asserting the recursive form, the opportunity is lost to illustrate to the reader *why* a recursive form is optimal. That is, what is it in the structure of the problem which allows a recursive form to emerge?

The remainder of this section, which borrows from the ideas and structure of [333], is divided into three parts:

1. Indirect estimation,
2. Uncorrelated measurements, and
3. Recursive estimation.

Although there are insights to be gained by understanding the derivation in detail, the reader may choose to skip the following on first reading.

I. Indirect Estimation

Suppose we are given the following problem:

$$\begin{array}{ll} \underline{m} = C\underline{z} + \underline{v} & \underline{v} \sim \mathcal{N}(\underline{0}, R) \\ \underline{q} = D\underline{z} + \underline{w} & \underline{w} \sim \mathcal{N}(\underline{0}, S) \end{array} \qquad \underline{z} \sim \mathcal{N}(\underline{0}, P). \tag{4.33}$$

That is, $\underline{m}$ is a known measurement of $\underline{z}$ and $\underline{q}$ is unknown, where the noise processes $\underline{v}, \underline{w}$ are uncorrelated with each other and with $\underline{z}$. We know the forms of the static estimators:

$$\hat{\underline{z}}(\underline{m}) = \Lambda_{zm}\Lambda_m^{-1}\underline{m} \qquad \hat{\underline{q}}(\underline{m}) = \Lambda_{qm}\Lambda_m^{-1}\underline{m}. \tag{4.34}$$

The required cross-statistics are easily derived:

$$\Lambda_{zm} = E\left[\underline{z}\,\underline{m}^T\right] = E\left[\underline{z}(C\underline{z}+\underline{v})^T\right] = PC^T \tag{4.35}$$

$$\Lambda_{qm} = E\left[\underline{q}\,\underline{m}^T\right] = E\left[(D\underline{z}+\underline{w})(C\underline{z}+\underline{v})^T\right] = DPC^T \tag{4.36}$$

$$\Lambda_q = E\left[\underline{q}\,\underline{q}^T\right] = E\left[(D\underline{z}+\underline{w})(D\underline{z}+\underline{w})^T\right] = DPD^T + S. \tag{4.37}$$

Thus the indirect estimator may be expressed as

$$\hat{\underline{q}}(\underline{m}) = DPC^T\Lambda_m^{-1}\underline{m} = D\hat{\underline{z}}(\underline{m}). \tag{4.38}$$

Similarly, the indirect estimation error covariance obeys

$$\tilde{P}_q = \Lambda_q - \Lambda_{qm}\Lambda_m^{-1}\Lambda_{qm}^T \tag{4.39}$$

$$= DPD^T + S - DPC^T\Lambda_m^{-1}CPD^T \tag{4.40}$$

$$= D\left(P - PC^T\Lambda_m^{-1}CP\right)D^T + S = D\tilde{P}_z D^T + S. \tag{4.41}$$

Thus we have the elegant conclusion that if we have an estimator $\hat{\underline{z}}$ for $\underline{z}$, then the estimator for any linear function of $\underline{z}$ is just that linear function of $\hat{\underline{z}}$:

$$\begin{array}{l} \underline{q} = D\underline{z} + \underline{w} \\ \underline{w} \sim (\underline{0}, S) \end{array} \qquad \Longrightarrow \qquad \begin{array}{r} \hat{\underline{q}}(\underline{m}) = D\hat{\underline{z}}(\underline{m}) \\ \tilde{P}_q = D\tilde{P}_z D^T + S. \end{array} \tag{4.42}$$

This relationship will be required in order to relate estimates at time $t+1$ in terms of estimates at time t.

II. Uncorrelated Measurements

Next, we wish to produce estimates based on two groups of measurements.

Suppose we wish to estimate zero-mean $\underline{z}$ based on measurements divided into two *uncorrelated* groups $\underline{m}_a, \underline{m}_b$. Note that we are making the rather unusual insistence that the *measurements* be uncorrelated, not just the measurement *errors*. That is, the statistics of the measurements obey

$$\Lambda_m = \text{cov}(\underline{m}) = \begin{bmatrix} \Lambda_a & \mathbf{0} \\ \mathbf{0} & \Lambda_b \end{bmatrix} \qquad \text{where } \underline{m} = \begin{bmatrix} \underline{m}_a \\ \underline{m}_b \end{bmatrix}. \tag{4.43}$$

Next, define the cross-statistics between the measurements and unknowns as

$$\Lambda_{zm} = E\left[\underline{z}\,\underline{m}^T\right] = E\left[\underline{z}\left[\underline{m}_a^T\ \underline{m}_b^T\right]\right] = \left[\Lambda_{za}\Lambda_{zb}\right]. \tag{4.44}$$

Then, using the standard LLSE form, we can derive the estimator for $\underline{z}$:

$$\hat{\underline{z}}(\underline{m}) = \Lambda_{zm}\Lambda_m^{-1}\underline{m} = \left[\Lambda_{za}\ \Lambda_{zb}\right]\begin{bmatrix} \Lambda_a & \mathbf{0} \\ \mathbf{0} & \Lambda_b \end{bmatrix}^{-1}\begin{bmatrix} \underline{m}_a \\ \underline{m}_b \end{bmatrix} \tag{4.45}$$

$$= \left[\Lambda_{za}\ \Lambda_{zb}\right]\begin{bmatrix} \Lambda_a^{-1} & \mathbf{0} \\ \mathbf{0} & \Lambda_b^{-1} \end{bmatrix}\begin{bmatrix} \underline{m}_a \\ \underline{m}_b \end{bmatrix} \tag{4.46}$$

$$= \Lambda_{za}\Lambda_a^{-1}\underline{m}_a + \Lambda_{zb}\Lambda_b^{-1}\underline{m}_b \tag{4.47}$$

$$= \hat{\underline{z}}(\underline{m}_a) + \hat{\underline{z}}(\underline{m}_b). \tag{4.48}$$

That is, if two measurement sets are uncorrelated, then the optimum estimator is just the *sum* of the estimates produced by each measurement set.

This result is important, in that we set $\underline{m}_a$ to be the entire past history of measurements, and $\underline{m}_b$ to be the current measurements at time t, which will thus lead to a recursive estimator

$$\hat{\underline{z}}(t|t) = \hat{\underline{z}}(\text{all past measurements}) + \hat{\underline{z}}(\text{measurements at time } t). \tag{4.49}$$

The measurements are normally *not* uncorrelated over time, therefore the key is to determine how to decorrelate them.

III. Recursive Estimation

Because (4.1) has the form of (4.33), it follows that

$$\hat{\underline{z}}(t|t-1) = A(t-1)\hat{\underline{z}}(t-1|t-1) \tag{4.50}$$

$$P(t|t-1) = A(t-1)P(t-1|t-1)A^T(t-1) + B(t-1)B^T(t-1). \tag{4.51}$$

This is known as the *prediction step* — the prediction over time of the estimates and related statistics.

In addition to predicting estimates over time, we also need to incorporate new measurements as they arrive, known as the *update step*. The key is the creation of an *innovations process* $\underline{\nu}(t)$:

$$\underline{\nu}(t) = \underline{m}(t) - \hat{\underline{m}}(t|t-1). \tag{4.52}$$

Although it may seem odd to construct estimates of measurements, $\underline{m}(t)$ is a random vector which obeys certain statistics and can be estimated, just like any other. From its definition (4.52), $\underline{\nu}(t)$ contains the information present in $\underline{m}(t)$ which cannot be inferred from past measurements $\underline{m}(0), \ldots, \underline{m}(t-1)$. That is, $\hat{\underline{m}}(t|t-1)$ is the predictable part of $\underline{m}(t)$, and $\underline{\nu}(t)$ contains only the *new* information in $\underline{m}(t)$.

Because $\underline{\nu}(t)$ is the estimation error in $\hat{\underline{m}}(t|t-1)$, by the orthogonality principle

$$E\left[\underline{\nu}(t)\,\underline{m}^T(s)\right] = \mathbf{0}, \quad 0 \le s < t. \tag{4.53}$$

That is, we have two *decorrelated* measurement sets: $\underline{\nu}(t)$ and $\underline{m}(0), \ldots, \underline{m}(t-1)$.

Next, because (4.4) also has the form of (4.33), again it follows that

$$\hat{\underline{m}}(t|t-1) = C(t)\hat{\underline{z}}(t|t-1) \tag{4.54}$$

thus the innovations can be written as

$$\underline{\nu}(t) = \underline{m}(t) - \hat{\underline{m}}(t|t-1) = C(t)\left[\underline{z}(t) - \hat{\underline{z}}(t|t-1)\right] + \underline{v}(t) \tag{4.55}$$

$$P_\nu(t) = C(t)P(t|t-1)C^T(t) + R(t). \tag{4.56}$$

Finally, let $\underline{e}(t)$ represent the predicted estimation error

$$\underline{z}(t) = \hat{\underline{z}}(t|t-1) + \underline{e}(t). \tag{4.57}$$

By orthogonality, $\underline{e}(t)$ must also be uncorrelated with $\underline{m}(s), 0 \leq s < t$. Therefore

$$\hat{\underline{z}}(t|t) = \hat{\underline{z}}(t|t-1) + \hat{\underline{e}}(t|t) \tag{4.58}$$

$$= \hat{\underline{z}}(t|t-1) + \hat{\underline{e}}\big(t|\underline{m}(0), \ldots, \underline{m}(t-1), \underline{\nu}(t)\big) \tag{4.59}$$

$$= \hat{\underline{z}}(t|t-1) + \hat{\underline{e}}\big(t|\underline{m}(0), \ldots, \underline{m}(t-1)\big) + \hat{\underline{e}}\big(t|\underline{\nu}(t)\big) \tag{4.60}$$

$$= \hat{\underline{z}}(t|t-1) + \underline{0} + \hat{\underline{e}}\big(t|\underline{\nu}(t)\big) \tag{4.61}$$

$$= \hat{\underline{z}}(t|t-1) + P_{e\nu}P_\nu^{-1}\underline{\nu}(t), \tag{4.62}$$

where (4.58) follows from I, (4.60) follows from II, (4.61) from orthogonality, and (4.62) from static estimation.

All that remains to complete the derivation is to find the innovation statistics:

$$P_\nu(t) = C(t)P(t|t-1)C^T(t) + R(t) \tag{4.63}$$

$$P_{e\nu}(t) = E\left[\underline{e}(t)\big(\underline{m}(t) - \hat{\underline{m}}(t|t-1)\big)^T\right] \tag{4.64}$$

$$= E\left[\underline{e}(t)\big(C(t)\underline{e}(t) + \underline{v}(t)\big)^T\right] \tag{4.65}$$

$$= P(t|t-1)C^T(t), \tag{4.66}$$

where the last relationship follows because the prediction error $\underline{e}(t)$ is uncorrelated with the measurement error $\underline{v}(t)$.

Thus we have completed the derivation of the *update step*:

$$\begin{aligned}\hat{\underline{z}}(t|t) &= \text{Predicted Estimate} + \text{Measurement Relevance} \cdot \quad \textit{New}\ \text{Information} \\ &= \quad \hat{\underline{z}}(t|t-1) \quad + \quad K(t) \quad \cdot \big(\underline{m}(t) - C(t)\hat{\underline{z}}(t|t-1)\big)\end{aligned} \tag{4.67}$$

where, from (4.63) and (4.66),

$$K(t) = P_{e\nu}P_\nu^{-1} = P(t|t-1)C^T(t)\left(C(t)P(t|t-1)C^T(t) + R(t)\right)^{-1}. \tag{4.68}$$

The corresponding estimation error covariance is

$$\begin{aligned}P(t|t) &= \text{Predicted Uncertainty} - \text{Uncertainty Reduction due to Measurements} \\ &= \quad P(t|t-1) \quad - \quad P_{e\nu}P_\nu^{-1}P_{e\nu}^T \\ &= \quad P(t|t-1) \quad - \quad K(t)C(t)P(t|t-1).\end{aligned} \tag{4.69}$$

The resulting discrete-time Kalman filter is summarized in Algorithm 1.

Algorithm 1 The Kalman Filter

Goals: Iteratively find estimates $\underline{\hat{z}}$ over $1 \leq t \leq \tau$, given initialization $(\underline{z}_o, P_o)$
Function $[\underline{\hat{z}}(1|1), P(1|1), \ldots, \underline{\hat{z}}(\tau|\tau), P(\tau|\tau)] = \mathbf{KF}(A, B, m, C, R, P_o, \underline{z}_o, \tau)$

$\underline{\hat{z}}(0|0) \leftarrow \underline{z}_o$ *Initialize at Time* $t = 0$
$P(0|0) \leftarrow P_o$
for $i \leftarrow 1 : \tau$ **do**
  $\underline{\hat{z}}(i|i-1) \leftarrow A\underline{\hat{z}}(i-1|i-1)$ *State Prediction*
  $P(i|i-1) \leftarrow AP(i-1|i-1)A^T + BB^T$ *Covariance Prediction*
  $K \leftarrow P(i|i-1)C^T \mathbf{inv}\big(CP(i|i-1)C^T + R\big)$ *Gain*
  $\underline{\hat{z}}(i|i) \leftarrow \underline{\hat{z}}(i|i-1) + K\big(m(i) - C\underline{\hat{z}}(i|i-1)\big)$ *State Update*
  $P(i|i) \leftarrow P(i|i-1) - KCP(i|i-1)$ *Covariance Update*
end for

IV. Discussion

The preceding derivation shows that the optimum linear least-squares estimator for a dynamic model of the form (4.1),(4.4) can be written in an efficient, recursive form, known as the *Kalman filter*. The three key steps are

I. Indirect estimation — how to generate estimates of linear functions of $\underline{z}$, allowing estimates to be projected or predicted over time;

II. Multiple measurements — how to generate estimates based on multiple groups of decorrelated measurements;

III. Measurement decorrelation — how to actually decorrelate the measurements in order to separate "past" and "current", the key to the recursion.

The structure of the resulting equations can be seen much more clearly in the time-invariant case:

$$A(t) = A \qquad B(t) = B \qquad C(t) = C \qquad R(t) = R, \tag{4.72}$$

in which the Kalman filter may be written as

Prediction:
$$\underline{\hat{z}}(t|t-1) = A\underline{\hat{z}}(t-1|t-1) \tag{4.73}$$
$$P(t|t-1) = AP(t-1|t-1)A^T + BB^T \tag{4.74}$$

Update:
$$\underline{\hat{z}}(t|t) = \underline{\hat{z}}(t|t-1) + K(t)\big(\underline{m}(t) - C\underline{\hat{z}}(t|t-1)\big) \tag{4.75}$$
$$P(t|t) = P(t|t-1) - K(t) \cdot C \cdot P(t|t-1) \tag{4.76}$$
$$K(t) = P(t|t-1) \cdot C^T \cdot \big(CP(t|t-1)C^T + R\big)^{-1} \tag{4.77}$$

As should be expected, the dynamics A, B affect only the prediction step; the update step is essentially a static estimator, taking place at a fixed point in time, and knowing

Example 4.2: Kalman Filtering and Interpolation

Suppose that we have a dynamic interpolation problem

$$\begin{array}{rl} z(t) & = \alpha\, z(t-1) + \beta\, w(t) \\ m(t) & = \quad z(t) \quad\quad + \quad v(t) \end{array} \qquad \text{where } w \sim \mathcal{N}(0,1),\ v \sim \mathcal{N}(0,\sigma^2) \tag{4.70}$$

Then the Kalman filter follows easily from (4.73)–(4.77):

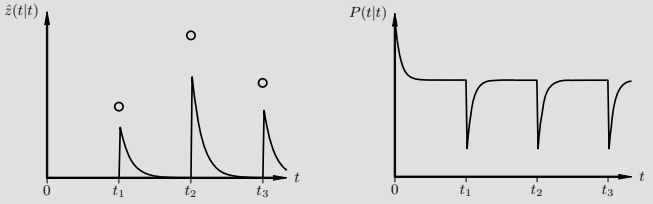
$$\begin{array}{lll} \text{Predict:} & \hat{z}(t|t-1) & = \alpha \hat{z}(t-1|t-1) \\ & P(t|t-1) & = \alpha^2 P(t-1|t-1) + \beta^2 \\ \\ \text{Update:} & K(t) & = P(t|t-1)/\big(P(t|t-1)+\sigma^2\big) \\ & \hat{z}(t|t) & = \hat{z}(t|t-1) + K(t)\big(m(t) - \hat{z}(t|t-1)\big) \\ & P(t|t) & = P(t|t-1)\big(1-K(t)\big) \end{array} \tag{4.71}$$

where it is important to observe that parameters α, β from the dynamic model appear *only* in the predict step, whereas the measurement parameter σ appears only in the update.

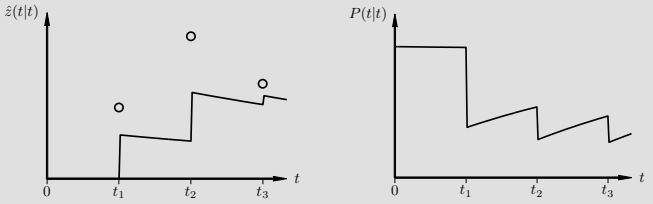
Suppose we have three measurements, plotted as circles, at times t_1, t_2, t_3. Let's consider the effects of varying α, β, and σ.

If $\alpha < 1$ then, over time, $\hat{z} \to 0$ and the rate of increase in P is mostly controlled by β. At each discrete measurement, $\hat{z}$ is updated as P is reduced:

$\alpha = 0.1$
$\beta = 1$
σ small



$\alpha = 0.9$
$\beta = \frac{1}{2}$
σ large



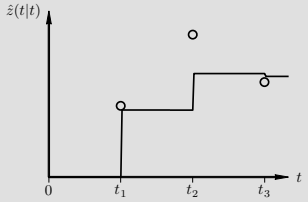
The more rapidly P grows over time, the more rapidly a past measurement is "forgotten." At any point in time where a measurement appears, the influence of the measurement on the estimate depends on the relative values of $P(T|t-1)$ and σ^2.

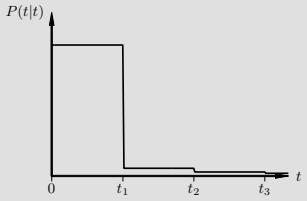
Example continues ...

Example 4.2: Kalman Filtering and Interpolation (cont'd)

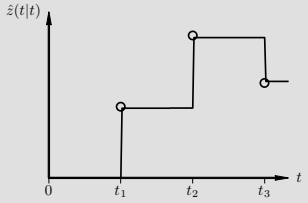
If $\alpha = 1$ then $\hat{z}$ is constant over time. The degree to which past measurements influence the current estimate depends on β:

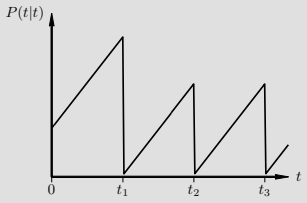
$\alpha = 1$
$\beta = 0$
σ small





$\alpha = 1$
$\beta = 1$
σ small





Clear patterns emerge:

Phenomenon	**Controlled By**
Rate of decay of $\hat{z}$	α
Converged value of P	α, β
Rate of growth of P	Mostly β
Closeness of $\hat{z}$ to m	σ^2 and $P(t\|t-1)$

nothing about dynamics. Similarly the measurements $\underline{m}$ and related model parameters C, R appear only in the update step. Note that if there are *no* measurements at some time t, then the update step is skipped completely:

$$\underline{\hat{z}}(t|t) = \underline{\hat{z}}(t|t-1) \qquad P(t|t) = P(t|t-1) \tag{4.78}$$

Understanding the connection to static estimation, we see that the update step is essentially written as *Form I* (3.65). Through the ABCD lemma in Appendix A.2, *Form II* (3.67) gives us an alternate version of the Kalman update

$$K(t) = P(t|t)C^T(t)R^{-1}(t) \tag{4.79}$$

$$P(t|t) = \left(P(t|t-1)^{-1} + C^T(t)R^{-1}(t)C(t)\right)^{-1}. \tag{4.80}$$

Finally, observe that the uncertainty P can *never* increase in the update step; that is,[2] $P(t|t) \leq P(t|t-1)$. The measurements can only improve our understanding; at worst the measurements are irrelevant, in which case the gain $K(t) = 0$ and the uncertainty $P(t|t) = P(t|t-1)$ is unchanged. On the other hand, the time-dynamics normally include an unpredictable stochastic term $\underline{w}(t)$, so the uncertainty P *does* increase by an amount BB^T in the prediction step.

Because the Kalman filter involves matrix–matrix multiplication and matrix inversion, the computational complexity of the filter is $\mathcal{O}(n^3)$ per iteration, where n is the dimensionality of $\underline{z}$. In cases where $\underline{z}$ represents some dynamic system with a few degrees of freedom, the Kalman filter is easily implemented in real-time. However, if $\underline{z}$ represents an image sequence, such as in video processing, where each image contains one million pixels, then $\mathcal{O}(n^3)$ may be prohibitive and other approaches need to be considered, which are the focus of the large-scale Kalman filtering methods in Chapter 10.

4.3 Kalman Filter Variations

A great many variations and forms of the Kalman filter have been developed to address different concerns or criticisms of the basic algorithm derived in Section 4.2. The most obvious concerns are the causality of the computed estimates, the numerical robustness or stability of the algorithm, the computational complexity, and the assumptions of Gaussianity and linearity. Each of these is dealt with in turn:

ESTIMATE CAUSALITY: The estimate $\hat{\underline{z}}(t|t)$ is an estimate of z at time t, based on measurements up to time t. The causal nature of the estimates can clearly be seen in Example 4.2. Because most dynamic processes are strongly correlated over time, measurements at times $t+1, t+2, \ldots$ would be useful in estimating $z(t)$:

$\Longrightarrow$ Section 4.3.3: Kalman Filter Smoothing
$\Longrightarrow$ Section 4.3.3: Fixed-lag Smoothing

NUMERICAL STABILITY: The numerical stability of the Kalman filter primarily involves the robustness in the calculation of the estimation error covariances, in particular whether the covariances remain positive-definite.[3] In particular, the covariance update (4.76)

$$P(t|t) = P(t|t-1) - K(t) \cdot C \cdot P(t|t-1) \tag{4.81}$$

[2] Matrix inequalities are *always* interpreted in a positive-definite sense. Thus the inequality $P(t|t) \leq P(t|t-1)$ really means $P(t|t-1) - P(t|t) \geq \mathbf{0}$, that is, that the difference is positive-semidefinite (see Appendix A.3).

[3] In estimation, if a prior covariance is used which fails to be positive-definite, even to only a tiny degree, it can lead to invalid estimates and negative estimation error variances.

looks suspicious, because $P(t|t)$ must be symmetric, yet the right-hand side of (4.81) looks asymmetric:

$\Longrightarrow$ Section 4.3.1: Joseph Stabilized Form

In cases where the conditioning of the dynamic problem is poor, it may be necessary to develop a different algorithm with improved conditioning:

$\Longrightarrow$ Section 4.3.1: Square Root Kalman Filter

COMPUTATIONAL AND STORAGE COMPLEXITY: The regular Kalman filter has a complexity of $\mathcal{O}(n^3)$ per iteration, for a state of n elements. For large problems this complexity is prohibitive; for example, a 1000×1000 dynamic image corresponds to $n = 10^6$. If a problem is stationary over time, then matrices $K(t), P(t|t), P(t|t-1)$ do not change over time and need to be computed only once:

$\Longrightarrow$ Section 4.3.2: Steady-State Kalman Filtering

In the event that the problem is too large to allow representation via large covariances, then the problem needs to be transformed or represented differently:

$\Longrightarrow$ Chapter 5: Modeling of Large Problems
$\Longrightarrow$ Chapter 8: Changes and Reductions of Bases
$\Longrightarrow$ Chapter 10: Kalman Filters for Large Problems

MODEL LINEARITY AND GAUSSIANITY: Many dynamic problems may be subject to nonlinear dynamics, or may be observed in the presence of non-Gaussian noise. If the dynamics are smooth, then repeated linearization may be appropriate:

$\Longrightarrow$ Section 4.3.4: Extended Kalman Filter

Where a linearization may be inadequate, the statistics may be better preserved by running a set of Kalman filters in parallel:

$\Longrightarrow$ Section 4.3.4: Unscented Kalman Filter
$\Longrightarrow$ Section 4.3.4: Ensemble Kalman Filter

Where the nonlinearities are severe, where the noise is highly non-Gaussian, or where it is undesirable to calculate the problem statistics, we may want an entirely implicit, nonparametric filter:

$\Longrightarrow$ Section 4.3.4: Particle Filter

4.3.1 Kalman Filter Algorithms

This section summarizes five of the most common implementation alternatives to the basic Kalman filter. However, like the classic Kalman filter, none of these are particularly well suited for huge estimation problems; specific Kalman filter variations for the multidimensional case are discussed in Chapter 10.

Update–Update Form

Some people consider it confusing, or unnecessarily complicated, to think of separate prediction and update steps. It is straightforward to substitute the prediction equations into the update ones in order to write $\hat{\underline{z}}_u(t) = \hat{\underline{z}}(t|t)$ directly in terms of $\hat{\underline{z}}_u(t-1) = \hat{\underline{z}}(t-1|t-1)$:

$$\hat{\underline{z}}_u(t) = \big(I - K(t)C(t)\big)A(t-1)\hat{\underline{z}}_u(t-1) + K(t)\underline{m}(t) \tag{4.82}$$

$$P_u(t) = \big(I - K(t)C(t)\big)\big(A(t-1)P_u(t-1)A^T(t-1) + B(t-1)B^T(t-1)\big). \tag{4.83}$$

A similar predict–predict form can also be derived.

Information Form

The standard discrete-time Kalman filter is written in terms of estimation error covariances $P(t|t), P(t+1|t)$. It is possible, however, to formulate [8,151] the Kalman recursion in terms of matrix *inverses* $P^{-1}(t|t), P^{-1}(t+1|t)$. There are three reasons why a matrix-inverse representation might be desirable:

1. Matrix Sparsity:

   As we show in Chapters 5 and 6, there is a wide variety of prior models and constraints in which P is dense, but P^{-1} is sparse, implying that the inverse covariance, or matrix *information*, may in many cases be a more natural and efficient representation. Sparse methods in Kalman filtering are developed further in Chapter 10.

2. Matrix Approximation:

   Related to the preceding point on sparsity, for a given computational and storage complexity we can usually approximate P^{-1} more accurately than P.

3. Representation of Complete Uncertainty:

   Given a variance σ^2:

Perfect certainty	$\sigma^2 = 0$
Complete uncertainty	$\sigma^2 = \infty$

Whereas representing in terms of an *inverse* variance:

Perfect certainty	$\sigma^{-2} = \infty$
Complete uncertainty	$\sigma^{-2} = 0$

The degrees of certainty anticipated in a given problem context may motivate one form over the other, as numerical computation with ∞ may be inconvenient.

Suppose we let a state variable be defined as

$$\hat{\underline{z}}_i(t|s) \triangleq P^{-1}(t|s)\hat{\underline{z}}(t|s). \tag{4.84}$$

In the inverse-covariance case, the update step becomes straightforward:

$$\text{From (3.67)} \qquad P^{-1}(t|t) = P^{-1}(t|t-1) + C^T(t)R^{-1}(t)C(t) \tag{4.85}$$

$$\text{From (4.78)} \qquad \hat{\underline{z}}_i(t|t) = \hat{\underline{z}}_i(t|t-1) + C^T(t)R^{-1}(t)\underline{\hat{m}}(t), \tag{4.86}$$

whereas the simple prediction stem becomes much more complicated in the inverted context. Applying the ABCD lemma (A.39) to (4.74) yields

$$P^{-1}(t|t-1) = A^{-T}P^{-1}(t-1|t-1)A^{-1} - A^{-T}P^{-1}(t-1|t-1)A^{-1}B\cdot$$
$$\left[B^TA^{-T}P^{-1}(t-1|t-1)A^{-1}B + I\right]^{-1}B^TA^{-T}P^{-1}(t-1|t-1)A^{-1}. \tag{4.87}$$

Finally, applying this result to (4.73) results in the state prediction

$$\hat{\underline{z}}_i(t|t-1) = \Big\{I - A^{-T}P^{-1}(t-1|t-1)A^{-1}B\cdot$$
$$\left[B^TA^{-T}P^{-1}(t-1|t-1)A^{-1}B + I\right]^{-1}B^T\Big\}A^T\hat{\underline{z}}_i(t-1|t-1). \tag{4.88}$$

In practice, we may choose to avoid the complexity of this latter step by performing the prediction step in the usual way:

$$\begin{aligned}\hat{\underline{z}}_i(t|t-1) &= P^{-1}(t|t-1)\hat{\underline{z}}(t|t-1)\\ &= P^{-1}(t|t-1)A(t-1)\hat{\underline{z}}(t-1|t-1)\\ &= P^{-1}(t|t-1)A(t-1)P(t-1|t-1)\hat{\underline{z}}_i(t|t-1).\end{aligned} \tag{4.89}$$

Square Root Form

The information form, just discussed, develops a Kalman filter based on covariance inverses which may be of some interest in sparse contexts, but of limited significance

otherwise. Of *much* greater practical interest is the square root form [8, 151, 231], which offers numerical robustness, making the Kalman filter applicable in poorly-conditioned circumstances. As poor conditioning is common in large-state problems, which we expect to encounter in image and multidimensional processing, the square root form is particularly relevant to us.

The strength of the square root Kalman filter stems from the two following properties of matrix square roots (Appendix A.8):

1. If a large covariance matrix P is approximated in any way, $\bar{P} \approx P$, then it is extremely likely that the approximate matrix $\bar{P}$ fails to be positive-definite,
$$P > \mathbf{0} \quad \text{but} \quad \bar{P} \not\geq \mathbf{0}, \tag{4.90}$$
whereas given Γ, a square root of P, and an approximate square root $\bar{\Gamma}$, then
$$\Gamma\Gamma^T > \mathbf{0} \quad \text{and} \quad \bar{\Gamma}\bar{\Gamma}^T \geq \mathbf{0}. \tag{4.91}$$
That is, an approximated square root guarantees positive-semidefiniteness.

2. Given Γ, a square root of P, then
$$P = \Gamma\Gamma^T \quad \Longrightarrow \quad \kappa(\Gamma) = \sqrt{\kappa(P)}. \tag{4.92}$$
The number of floating-point digits required is roughly $\log_{10}(\kappa)$, therefore the square root form requires only *half* the precision of the standard Kalman filter, leading to significant savings in storage and computational complexity (also see Appendix A.8).

Thus the square root Kalman filter offers substantial benefits of numerical robustness.

One possible square root filter [333] follows from applying QR decompositions (Appendix A.7.3). Suppose we begin with $\underline{\hat{z}}(t|t-1), \Gamma(t|t-1)$ where
$$P(t|t-1) = \Gamma(t|t-1)\Gamma^T(t|t-1), \tag{4.93}$$
such that Γ is the numerically robust representation of covariance P. Then we can formulate square root update and prediction steps, as follows.

Square Root Update Step:

We begin with a QR decomposition
$$\underbrace{\begin{bmatrix} C(t)\Gamma(t|t-1) & R^{1/2}(t) \\ \Gamma(t|t-1) & \mathbf{0} \end{bmatrix}}_{\text{Given}\ldots} \longrightarrow \underbrace{\begin{bmatrix} Q(t) & \mathbf{0} \\ \bar{K}(t) & \Gamma(t|t) \end{bmatrix}}_{\text{Computed via QR}} \tag{4.94}$$
which computes the updated uncertainty $\Gamma(t|t)$; the updated estimates are found as
$$\underline{\hat{z}}(t|t) = \underline{\hat{z}}(t|t-1) + \bar{K}(t)Q^{-1}(t)\big(\underline{m}(t) - C(t)\underline{\hat{z}}(t|t-1)\big). \tag{4.95}$$

Square Root Predict Step:

The estimates are predicted as usual,

$$\underline{\hat{z}}(t+1|t) = A(t)\underline{\hat{z}}(t|t). \tag{4.96}$$

A second QR decomposition computes the predicted uncertainty:

$$\underbrace{\begin{bmatrix} A(t)\Gamma(t|t) & B(t) \end{bmatrix}}_{\text{Given } \ldots} \longrightarrow \underbrace{\begin{bmatrix} \Gamma(t+1|t) & \mathbf{0} \end{bmatrix}}_{\text{Computed via QR}} \tag{4.97}$$

Why Does this Work?

The QR decomposition (Appendix A.7.3) orthogonally transforms a given matrix to make it triangular. By squaring both sides the orthogonal matrix disappears

$$AU = H \quad \Longrightarrow \quad AA^T = HH^T. \tag{4.98}$$

Therefore each of (4.94) and (4.97) can be squared, and like terms equated, to validate the result. We begin with the simpler predict step: squaring (4.97) yields

$$A(t)\Gamma(t|t)\Gamma^T(t|t)A^T(t) + B(t)B^T(t) = \Gamma(t+1|t)\Gamma^T(t+1|t) + \mathbf{0}, \tag{4.99}$$

thus

$$A(t)P(t|t)A^T(t) + B(t)B^T(t) = P(t+1|t), \tag{4.100}$$

which we recognize from the usual Kalman filter prediction step (4.74).

Next, squaring the more difficult (4.97) yields

$$\begin{bmatrix} C\Gamma(t|t-1)\Gamma^T(t|t-1)C^T & C\Gamma(t|t-1)\Gamma^T(t|t-1) \\ \Gamma(t|t-1)\Gamma^T(t|t-1)C^T & \Gamma(t|t-1)\Gamma^T(t|t-1) \end{bmatrix} = \begin{bmatrix} QQ^T & Q\bar{K}^T \\ \bar{K}Q^T & \bar{K}\bar{K}^T + \Gamma(t|t)\Gamma^T(t|t) \end{bmatrix}. \tag{4.101}$$

Equating the top-left terms yields

$$QQ^T = CP(t|t-1)C^T + R. \tag{4.102}$$

Then, selecting either of the off-diagonal terms yields

$$\bar{K} = P(t|t-1)C^TQ^{-1}. \tag{4.103}$$

Finally, equating the lower-right terms then yields the desired result:

$$P(t|t-1) = P(t|t) + \bar{K}\bar{K}^T \tag{4.104}$$
$$= P(t|t) + P(t|t-1)C^T Q^{-1} Q^{-T} C P(t|t-1) \tag{4.105}$$
$$= P(t|t) + P(t|t-1)C^T \big(CP(t|t-1)C^T + R\big)^{-1} CP(t|t-1), \tag{4.106}$$

which is the Kalman filter covariance update (4.76).

A great many variations on the square root filter are available [151], depending on exactly what sort of matrix factorization is undertaken, and whether the square root filter is in regular, predict–predict, update–update, or information form.

Joseph Stabilized

The most common form for the updated error covariance (4.69), (4.76) is

$$P(t|t) = \big(I - K(t)C(t)\big)P(t|t-1). \tag{4.107}$$

Given that $P(t|t)$ is a covariance matrix, by definition it must be symmetric and positive-semidefinite. However, numerical rounding errors in the computation of $K(t)$, or of the products in (4.107), can lead to the computed $P(t|t)$ being asymmetric or containing negative eigenvalues.

An alternative, but algebraically equivalent, form of (4.107) is the *Joseph stabilized form* [47]:

$$P(t|t) = \big(I - K(t)C(t)\big)P(t|t-1)\big(I - K(t)C(t)\big)^T + K(t)R(t)K^T(t). \tag{4.108}$$

In general, because

$$P \geq \mathbf{0} \implies APA^T \geq \mathbf{0} \ \ \forall A, \tag{4.109}$$

then if $P(t|t-1)$ and $R(t)$ are symmetric, positive-semidefinite, then the Joseph form (4.108) guarantees that $P(t|t)$ remains symmetric, positive-semidefinite, regardless of rounding errors in K.

The improvement in numerical stability and conditioning comes at a cost of additional matrix products in (4.108) relative to (4.107).

Singular Filtering

There may be some circumstances under which one or more measurements may be known *exactly*, such that $R(t)$ is singular, meaning that $\big(CP(t|t-1)C^T + R\big)$ may not be invertible, as required in the update step.

The general approach [8], then, is to divide the problem into two parts:

1. The part which is measured exactly, and therefore does not need to be estimated.
2. The part which is measured, as usual, in the presence of additive noise, which is estimated using the regular Kalman filter.

In particular, suppose that the measurement model can be written as

$$\underline{m}(t) = \begin{bmatrix} \underline{m}_1(t) \\ \underline{m}_2(t) \end{bmatrix} = \begin{bmatrix} C_1(t) \\ C_2(t) \end{bmatrix} \underline{z} + \begin{bmatrix} \underline{0} \\ \underline{v}_2(t) \end{bmatrix} \qquad \underline{v}_2 \sim R_2(t). \tag{4.110}$$

Without loss of generality, we assume C_1 to have full row rank,[4] so that we can similarly transform the state $\underline{z}$ into two corresponding parts,

$$\begin{bmatrix} \underline{z}_1(t) \\ \underline{z}_2(t) \end{bmatrix} = D\underline{z} = \begin{bmatrix} C_1(t) \\ D_2 \end{bmatrix} \underline{z}. \tag{4.111}$$

D_2 is any matrix which makes D invertible, guaranteeing that $\underline{z}_2$ spans the entire portion of the state not exactly measured by $\underline{m}_1$. The first part of the state is known exactly,

$$\hat{\underline{z}}_1(t) = \underline{m}_1(t) \tag{4.112}$$

and for the second part an estimation problem can be defined:

$$\underline{m}_2(t) = C_2(t)\underline{z}(t) + \underline{v}_2(t) = C_2(t) \begin{bmatrix} C_1(t) \\ D_2 \end{bmatrix}^{-1} \begin{bmatrix} \underline{m}_1(t) \\ \underline{z}_2(t) \end{bmatrix} + v_2(t) \tag{4.113}$$

$$\operatorname{cov}\big(\underline{z}_2(t)\big) = D_2 P(t|t-1) D_2^T, \tag{4.114}$$

from which we estimate $\hat{\underline{z}}_2(t)$ with estimation error covariance $\tilde{P}_2(t)$. The estimates for the partitioned state allow us to reconstruct the original:

$$\hat{\underline{z}}(t) = \begin{bmatrix} C_1(t) \\ D_2 \end{bmatrix}^{-1} \begin{bmatrix} \hat{\underline{z}}_1(t) \\ \hat{\underline{z}}_2(t) \end{bmatrix} \qquad P(t|t) = \begin{bmatrix} C_1(t) \\ D_2 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \tilde{P}_2(t) \end{bmatrix} \begin{bmatrix} C_1(t) \\ D_2 \end{bmatrix}^{-T}. \tag{4.115}$$

4.3.2 Steady-State Kalman Filtering

The case of linear time-invariant (LTI) systems is of special interest. Although such systems may occur infrequently in practice, their time-stationarity leads to attractive simplifying properties.

Suppose we are given the usual first-order Gauss–Markov model, as in (4.1), but now time-invariant:

[4] If there are redundant exact measurements, the redundant ones can be removed until full rank is achieved.

$$\underline{z}(t+1) = A\underline{z}(t) + B\underline{w}(t) \qquad \underline{m}(t) = C\underline{z}(t) + \underline{v}(t) \qquad \underline{v} \sim R, \tag{4.116}$$

where time-invariance implies that model parameters A, B, C, R are fixed over time.

If the dynamic system is stable, meaning that the eigenvalues λ_i of A satisfy $|\lambda_i| < 1$, then the statistics of $\underline{z}$ will reach steady-state (that is, will be invariant over time). In steady-state the covariance $P(t)$ of $\underline{z}$ will converge to the solution of the algebraic Lyapunov equation [200]

$$P_\infty = AP_\infty A^T + BB^T. \tag{4.117}$$

The Kalman filter itself can *also* reach steady-state, meaning that the Kalman gain $K(t)$ and the estimation error covariances $P(t|t-1), P(t|t)$ converge to fixed values as $t \to \infty$. In particular, under fairly general conditions (system reachability and observability[5]), the predicted error covariance will converge to the unique, positive-definite solution to the algebraic Riccati equation [200]

$$P_p = AP_pA^T + BB^T - AP_pC^T\left(CP_pC^T + R\right)^{-1}CP_pA^T, \tag{4.118}$$

from which the constant Kalman gain follows as

$$K = P_pC^T\left(CP_pC^T + R\right)^{-1}. \tag{4.119}$$

Finally, it is instructive to see how the predicted estimation error evolves; given

$$\underline{z}(t+1) = A\underline{z}(t) + B\underline{w}(t) \tag{4.120}$$

$$\hat{\underline{z}}(t+1|t) = A\hat{\underline{z}}(t|t) = A\Big(K\big(\underline{m}(t) - C\hat{\underline{z}}(t|t-1)\big) + \hat{\underline{z}}(t|t-1)\Big), \tag{4.121}$$

then the predicted estimation error is

$$\tilde{\underline{z}}(t+1|t) = \hat{\underline{z}}(t+1|t) - \underline{z}(t+1) \tag{4.122}$$

$$= A\Big(K\big(\underline{m}(t) - C\hat{\underline{z}}(t|t-1)\big) + \hat{\underline{z}}(t|t-1)\Big) - \Big(A\underline{z}(t) + B\underline{w}(t)\Big) \tag{4.123}$$

$$= A\big(I - KC\big)\tilde{\underline{z}}(t|t-1) + AK\underline{v}(t) - B\underline{w}(t). \tag{4.124}$$

That is, the predicted estimation error itself obeys a dynamic relationship which is stable when $A(I-KC)$ is stable, a stability *not* dependent on the stability of A. Because the Kalman gain matrix K is essentially the regularized solution to the inverse problem $\underline{m} = C\underline{z}$, we expect that $KC \approx I$, and so $(I - KC) \approx 0$.

Given the steady-state gain matrix K, the Kalman filter reduces to

[5] Meaning that, over time, each element in $\underline{z}$ is affected by the driving process $\underline{w}$, and that the measurements feel the influence of each element of $\underline{z}$ to some degree; in other words, that no portion of the state be completely decoupled from $\underline{w}$ and $\underline{m}$. Specifically, it is required that $[B\ AB\ AAB\ldots]$ and $[C^T\ A^TC^T\ A^TA^TC\ldots]$ be full rank.

$$\underline{\hat{z}}(t|t-1) = A\underline{\hat{z}}(t-1|t-1) \tag{4.125}$$

$$\underline{\hat{z}}(t|t) = \underline{\hat{z}}(t|t-1) + K\big(\underline{m}(t) - C\underline{\hat{z}}(t|t-1)\big), \tag{4.126}$$

involving only fast matrix–vector operations, no matrix multiplications or inverses, a huge reduction in complexity to $\mathcal{O}(n^2)$ per iteration from $\mathcal{O}(n^3)$ per iteration for the full filter. Keep in mind the following limitations:

1. Steady-state filtering applies only to cases in which the underlying dynamics are stationary, *and* in which the measurement model is stationary (constant model C and quality R). In particular, cases of irregular measurements (e.g., sparse measurements of an image) do not satisfy stationarity.

2. The steady-state statistics and gain matrix can be difficult to compute; in particular, the Riccati equation (4.118) can be very challenging computationally. In practice, it is often much simpler to run the regular Kalman filter for t steps, where t is chosen "large enough" such that $K(t) \simeq K, P(t|t-1) \simeq P_p$.

4.3.3 Kalman Filter Smoother

The Kalman filter is strictly causal, in the sense that an estimate $\underline{\hat{z}}(t)$ is affected by measurements only at time t and earlier. However, even for causal dynamic processes, measurements from the future can be extremely useful in estimating the present.

For example, consider the simple first-order causal dynamic model

$$z(t+1) = \alpha z(t) + w(t). \tag{4.127}$$

The autocorrelation of z is easily computed as

$$E\big[z(t+s)z(t)\big] = E\big[\big(\alpha z(t+s-1) + w(t)\big)z(t)\big] = \alpha^s E\big[z(t)z(t)\big] \tag{4.128}$$

$$E\big[z(t-s)z(t)\big] = E\big[z(t-s)\big(\alpha z(t-1) + w(t)\big)\big] = \alpha^s E\big[z(t-s)z(t-s)\big], \tag{4.129}$$

where $s > 0$. If z is in statistical steady-state, meaning that $E[z(t)z(t)] = \xi$ is constant over time, then the autocorrelation of causal process $z(t)$ is perfectly symmetric over time,

$$E\big[z(t_1)z(t_2)\big] = \alpha^{|t_1 - t_2|}\xi, \tag{4.130}$$

implying that measurements of $z(t-1), z(t+1)$ are both *equally* useful in estimating $z(t)$. Although this example examined a simple, scalar, stationary case, the conclusion applies generally: a noncausal estimator is likely to yield more accurate results than a causal one. Indeed, as the process noise gets smaller (smaller B) and as the measurements become more infrequent, the greater is the possible benefit of future measurements in reducing the estimation error in the present.

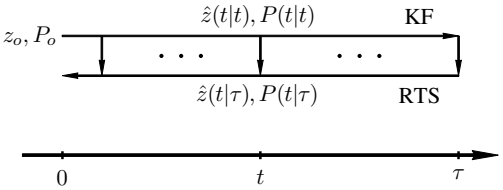


Fig. 4.2. The Kalman smoother is a two-pass algorithm: first the regular Kalman filter forwards over time, and then a backwards iteration. All of the estimated states and error covariances from the forward pass must be saved for the backward pass, meaning that the storage requirements are greatly increased over those of the standard Kalman filter alone.

The idea is sketched in Figure 4.2: given the results $\underline{\hat{z}}(t|t), P(t|t)$, $0 \leq t \leq \tau$ of the Kalman filter, is it possible to find an algorithm, ideally iterative, which works backwards, finding $\underline{\hat{z}}(t|\tau), P(t|\tau)$?

Indeed, such an algorithm, the Kalman smoother, was developed by Rauch, Tung, and Striebel [266]. Recall the innovations process $\underline{\nu}(t)$ from (4.52),

$$\underline{\nu}(t) = \underline{m}(t) - \underline{\hat{m}}(t|t-1). \tag{4.131}$$

This process is white, but contains all of the information present in the measurements. Then the smoothed estimate can be written [333] as the sum of the predicted estimate, $\underline{\hat{z}}(t|t-1)$, and the estimate of the prediction error $\underline{\tilde{z}}(t|t-1)$ based on future measurements:

$$\underline{\hat{z}}(t|\tau) = \sum_{s=0}^{\tau} K(t,s)\underline{\nu}(s) = \underline{\hat{z}}(t|t-1) + \sum_{s=t}^{\tau} K(t,s)\underline{\nu}(s) \tag{4.132}$$

$$= \underline{\hat{z}}(t|t-1) + \sum_{s=t}^{\tau} E\big[\underline{\tilde{z}}(t|t-1)\underline{\nu}^T(s)\big] E\big[\underline{\nu}(s)\underline{\nu}^T(s)\big]^{-1}\underline{\nu}(s) \tag{4.133}$$

$$= \underline{\hat{z}}(t|t-1) + \sum_{s=t}^{\tau} E\big[\underline{\tilde{z}}(t|t-1)\underline{\tilde{z}}^T(s|s-1)\big] C^T(s)\cdot \big(C(s)P(s|s-1)C^T(s) + R(s)\big)^{-1}\underline{\nu}(s) \tag{4.134}$$

$$= \underline{\hat{z}}(t|t-1) + P(t|t-1)\sum_{s=t}^{\tau} F^T(t)\ldots F^T(s-1)C^T(s)\cdot \big(C(s)P(s|s-1)C^T(s) + R(s)\big)^{-1}\underline{\nu}(s), \tag{4.135}$$

where

$$F(t) = A(t)\big\{I - K(t)C(t)\big\} \tag{4.136}$$

is the dynamic matrix, as in (4.124), governing the evolution of the prediction errors. By inspection, the summation term in (4.135) admits a recursive form, leading to the

backwards recursion for the Kalman smoother:

$$\underline{\hat{z}}(t|\tau) = \underline{\hat{z}}(t|t-1) + P(t|t-1)\underline{\hat{u}}(t|\tau) \tag{4.137}$$

where

$$\underline{\hat{u}}(t-1|\tau) = F^T(t-1)\underline{\hat{u}}(t|\tau) + C^T(t-1)\cdot \\ \big(C(t-1)P(t-1|t-2)C^T(t-1) + R(t-1)\big)^{-1}\underline{\nu}(t-1) \tag{4.138}$$

with the recursion initialized as $\underline{\hat{u}}(\tau+1|\tau) = \underline{0}$.

Although this recursion was relatively straightforward to derive, there are simpler and more elegant alternatives [151]:

$$\underline{\hat{z}}(t|\tau) = \underline{\hat{z}}(t|t) + Q(t)\big(\underline{\hat{z}}(t+1|\tau) - \underline{\hat{z}}(t+1|t)\big) \tag{4.139}$$

$$P(t|\tau) = P(t|t) + Q(t)\big(P(t+1|\tau) - P(t+1|t)\big)Q^T(t), \tag{4.140}$$

where

$$Q(t) = P(t|t)A^T(t)P^{-1}(t+1|t). \tag{4.141}$$

The application of these latter smoother equations to one-dimensional first- and second-order systems is illustrated in Example 4.3.

Although it is clear that the smoother can provide improvements in estimation accuracy by using future measurements, it is important to understand its limitations:

- The Kalman filter is recursive, requiring no past history to be stored. However, because the smoother requires the entire forward sequence $\underline{\hat{z}}(t|t-1), P(t|t-1)$ to be stored, the storage complexity of the smoother is τ times greater than that of the Kalman filter.

  In those cases where $\underline{z}$ is a long vector, such as in image estimation, this storage burden may be significant. This issue will be examined further in Chapter 10, and the reader may be interested in looking at Example 10.1 on page 332, where the Kalman smoother is applied to an image.

- The delay between receiving measurement $\underline{m}(0)$ and producing the smoothed estimate $\underline{\hat{z}}(0|\tau)$ clearly grows with τ. However (see Problem 4.3), although $P(0|\tau)$ is a decreasing function of τ, in most circumstances the benefit is limited to small values of τ, where larger τ increases delay and storage requirements but with almost no additional benefit in accuracy.

  In those cases where a very modest smoothing window is adequate, it is straightforward to implement a recursive fixed-lag smoother directly in the Kalman filter. Indeed, we just define an augmented state, as in (4.8):

Example 4.3: Recursive Smoothing and Interpolation

Example 4.2 on page 98 looked at the application of the Kalman filter to interpolation. Recall that the estimates $\hat{z}(t|t)$ were conspicuously causal:

$$\text{Measurement } m(s) \text{ influences } \hat{z}(t|t) \text{ only for } t \geq s.$$

However, the causality of the Kalman filter does not imply a causality in the underlying statistics: $m(s)$ *does* tell us a great deal about $z(t)$ for $t < s$. That is, we expect quite different results between $\hat{z}(t|t)$ from the Kalman filter and $\hat{z}(t|\tau)$ from the Kalman smoother.

We will consider two models:

I. First Order: $z(t) = z(t-1) + w(t-1)$ — z is a random walk

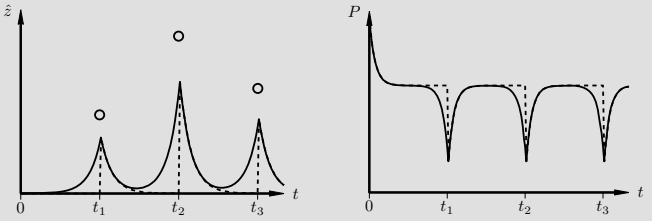
II. Second Order: $z(t) = z(t-1) + \Delta(t-1)$, $\Delta(t) = \Delta(t-1) + w(t-1)$ — In this case the *slope* of z is a random walk

where we can rewrite the second-order model in a single state as

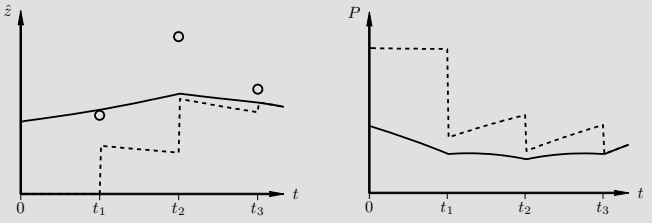
$$\underline{z}(t) = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \underline{z}(t-1) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} w(t-1). \tag{4.142}$$

The following two results are for the first-order case, where the difference between the causal estimates of the Kalman filter (dashed) and the Kalman smoother (solid) can be clearly seen:

$\alpha = 0.1$
$\beta = 1$
σ small



$\alpha = 0.9$
$\beta = \frac{1}{2}$
σ large

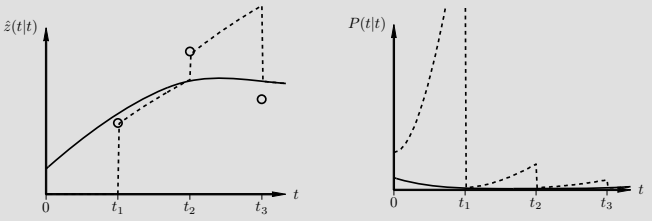


Example continues . . .

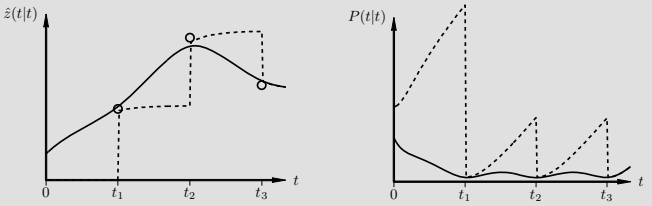
Example 4.3: Recursive Smoothing and Interpolation (cont'd)

Two further results for the second-order case:

$\alpha = 0.9$
$\beta = 0.1$
σ small



$\alpha = 0.3$
$\beta = 0.9$
σ small



Observations:

- Although the dynamic model for $z(t)$ is written causally, the smoothed estimates $\hat{z}(t|\tau)$ are *acausally* dependent on the measurements.
- In all cases, the smoother can only improve our certainty,

$$P(t|\tau) \leq P(t|t)$$

since the measurements available to the smoother include all of those available to the filter, plus some more.

- The smoothed estimates are, in most cases, more appealing, aesthetic, and desirable than the causal ones. The difference is particularly striking in the second-order case.

$$\underline{\bar{z}}(t) \triangleq \begin{bmatrix} \underline{z}(t) \\ \underline{z}(t-1) \\ \vdots \\ \underline{z}(t-\tau) \end{bmatrix} = \begin{bmatrix} A(t-1) & & \mathbf{0} \\ I & & \\ & \ddots & \vdots \\ \mathbf{0} & I & \mathbf{0} \end{bmatrix} \underline{\bar{z}}(t-1) + \begin{bmatrix} B(t-1) \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix} \underline{w}(t-1) \tag{4.143}$$

such that the resulting estimates

$$\hat{\underline{z}}(t|t) = \begin{bmatrix} \hat{\underline{z}}(t|t) \\ \vdots \\ \hat{\underline{z}}(t-\tau|t) \end{bmatrix} \tag{4.144}$$

include the desired, fixed-lag smoothed estimate $\hat{\underline{z}}(t-\tau|t)$. Although the augmented dynamic matrix in (4.143) is sparse, the computed error covariances $P(t|t), P(t|t-1)$ will be dense. Therefore with a storage complexity of $\mathcal{O}(\tau^2)$ and a computational complexity of $\mathcal{O}(\tau^3)$ the fixed-lag approach is feasible for only relatively modest τ.

Further details on the smoother, including square root and information forms, can be found in [8, 28, 125, 151]. Multidimensional Kalman smoothers are discussed in Chapter 10.

4.3.4 Nonlinear Kalman Filtering

In this text we focus on linear estimation problems, so the following discussion on nonlinear filters represents a bit of a digression, included because the methods are fairly intuitive and nicely complement the Kalman filter discussion of the rest of this chapter.

Extended Kalman Filter

The extended Kalman filter [151, 156] allows for the system dynamics and measurement functions to be nonlinear functions of the current state. However, because nonlinear transformations of random vectors (Appendix B.2) are very complicated, the dynamics and measurement functions are linearized at each time step in computing the predicted and updated error covariances.

So, given possibly nonlinear dynamic and observation models

$$\underline{z}(t+1) = \mathcal{A}\big(\underline{z}(t)\big) + B\underline{w}(t) \qquad \underline{m}(t) = \mathcal{C}\big(\underline{z}(t)\big) + \underline{v}(t) \tag{4.145}$$

the Kalman filter equations of (4.73)–(4.77) may be rewritten as

Prediction:
$$\hat{\underline{z}}(t|t-1) = \mathcal{A}\big(\hat{\underline{z}}(t-1|t-1)\big) \tag{4.146}$$
$$P(t|t-1) = A_{t-1}P(t-1|t-1)A_{t-1}^T + BB^T \tag{4.147}$$

Update:
$$\hat{\underline{z}}(t|t) = \hat{\underline{z}}(t|t-1) + K(t)\Big(\underline{m}(t) - \mathcal{C}\big(\hat{\underline{z}}(t|t-1)\big)\Big) \tag{4.148}$$
$$P(t|t) = P(t|t-1) - K(t)\cdot C_t \cdot P(t|t-1) \tag{4.149}$$
$$K(t) = P(t|t-1)\cdot C_t^T \cdot \big(C_t P(t|t-1)C_t^T + R\big)^{-1}, \tag{4.150}$$

where A_t, C_t are the linearizations

$$A_t = \left.\frac{\partial \mathcal{A}}{\partial \underline{z}}\right|_{\underline{\hat{z}}(t|t)} \qquad C_t = \left.\frac{\partial \mathcal{C}}{\partial \underline{z}}\right|_{\underline{\hat{z}}(t|t-1)} \tag{4.151}$$

The implementation is thus a straightforward extension of the standard Kalman filter. As only the estimates themselves, and not the associated statistics, are propagated via the nonlinear dynamic and measurement functions, the performance of the extended Kalman filter can be quite poor, especially for rapidly-varying or discontinuous nonlinearities.

Ensemble Kalman Filter

The one limitation of the extended Kalman filter is the inability to propagate the statistics through the model nonlinearities. Although such nonlinear propagation is difficult to do analytically (see Section B.2), it is easy to do so numerically by computing sample statistics. Even for a linear problem, inferring the statistics from samples frees us from having to predict and update the estimation error covariances, convenient for dynamic problems having a large state.

Such a covariance propagation by sampling is the essence of the ensemble Kalman filter [100, 101, 164]. An ensemble of estimates is predicted at each time step, and then updated on the basis of measurements with added random noise.

We begin by initializing random samples from the prior distribution

$$\underline{\hat{z}}_i(0|-1) \sim \mathcal{N}\big(\underline{z}(0), P(0)\big). \tag{4.152}$$

Each of the random samples is essentially iterated as a Kalman filter, all in parallel. Each sample is passed through the (possibly) nonlinear prediction

$$\underline{\hat{z}}_i(t|t-1) = \mathcal{A}\big(\underline{\hat{z}}_i(t-1|t-1)\big) \tag{4.153}$$

from which the estimate and error covariance can be computed as sample statistics:

$$\underline{\hat{z}}(t|t-1) = \text{Sample Mean}\left(\{\underline{\hat{z}}_i\}\right), \tag{4.154}$$

$$P(t|t-1) = \text{Sample Covariance}\left(\{\underline{\hat{z}}_i\}\right). \tag{4.155}$$

For the update step, random noise $\underline{v}_i(t)$ is added to the measurements in order to prevent the ensemble members from converging:

$$\underline{\hat{z}}_i(t|t) = \underline{\hat{z}}_i(t|t-1) + K(t)\Big(\big(\underline{m}(t) + \underline{v}_i(t)\big) - C\underline{\hat{z}}(t|t-1)\Big) \tag{4.156}$$

$$K(t) = P(t|t-1) \cdot C^T \cdot \big(CP(t|t-1)C^T + R\big)^{-1}, \tag{4.157}$$

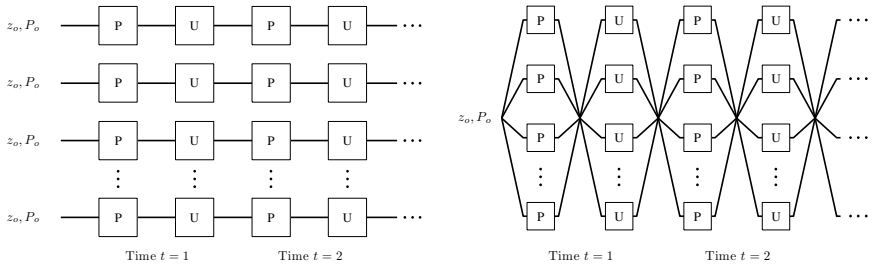


Fig. 4.3. The ensemble (left) and unscented (right) Kalman filters take very different approaches in the use of ensembles to handle nonlinearities. The ensemble KF has a fixed set of filters, run in parallel, whereas the unscented KF computes an explicit estimate $\underline{\hat{z}}$ at every step and resamples after every predict and update step.

where we observe the absence of the equation to compute $P(t|t)$, as this is no longer needed.

The drawback of the ensemble Kalman filter is that the number of ensemble members may need to be relatively large in order to obtain reasonable sample statistics of the estimation mean and covariance. As with the regular Kalman filter, a number of variations exist, such as a square root ensemble Kalman filter [311].

Unscented Kalman Filter

The unscented Kalman filter [181,325,326] is closely related to the ensemble Kalman filter, above, with two key differences:

1. In the ensemble Kalman filter we essentially had N individual Kalman filters, running in parallel, and not interacting with each other except in the computation of sample means and covariances.

   In contrast, the unscented Kalman filter is essentially a *single* filter, with a single predicted and estimated value per time step, which resamples *every* predict and update step, as illustrated in Figure 4.3.

2. Whereas the ensemble filter is a Monte Carlo method, requiring a substantial number of ensemble members to properly reproduce the underlying statistics, the unscented filter uses a small number of deterministic samples. Second, rather than the usual sample mean and sample covariance of (4.154), (4.155), a weighted version of these equations is used.

The propagation of statistics through nonlinearities is considerably improved by this method over the extended Kalman filter, and with many fewer ensemble members than the ensemble Kalman filter. However, we still have a second-order representation (mean and covariance) at each time step, a representation which may be poor in the case of highly asymmetric distributions stemming from sharp nonlinearities.

Particle Filters

Particle filters [170, 256] are a natural continuation of the ensemble and unscented approaches, however the topic is a large field, with entire textbooks [88, 272] dedicated to the topic, so we can provide only a superficial treatment here.

The extended, ensemble, and unscented filters all suffer from the explicit computation of and reliance on error covariances, essentially implicitly assuming a Gaussian distribution. Since most nonlinear problems will have asymmetric non-Gaussian statistics, the covariance representation is known to be poor.

The key to the particle filter is that the distribution of the estimated state is represented *implicitly*, rather than explicitly; we do not explicitly calculate an estimate mean or covariance.

Given a set of estimates $\{\underline{\hat{z}}_i(t-1|t-1)\}$ at time $t-1$, each estimate is imagined as a particle (or hypothesis) in n-dimensional space. One iteration of the particle filter proceeds as follows:

- Prediction: Apply the temporal dynamics, which could be linear or nonlinear, and deterministic or stochastic. Each particle is predicted separately, giving us the predicted set
$$\underline{\hat{z}}_i(t|t-1) = \text{Predict}\big(\underline{\hat{z}}_i(t-1|t-1)\big). \tag{4.158}$$
- Update: We assert some metric which measures the quality of each particle, as assessed by the measurements. Essentially, we want to infer the likelihood (or some equivalent metric) of each state particle
$$p_{t,i} = \Pr\big(\underline{\hat{z}}_i(t|t-1)|\underline{m}(t)\big). \tag{4.159}$$
- Resampling: We need to steer the ensemble set in favour of likely candidates; that is, we wish to probabilistically favour those states which are consistent with the measurements, however with some probability also preserving less likely states. The biasing towards the measurements ensures accurate estimation, with most states highly consistent with the observations, whereas the preservation of less likely states adds robustness, for example in the case where the measurement is a spurious outlier, or in cases of occlusion (where part of the state is hidden).

In principle, we could nonparametrically estimate the distribution $p\big(\underline{\hat{z}}(t)|\underline{m}(t)\big)$ from the likelihoods $\{p_{t,i}\}$, and then randomly resample particles from this distribution.

At every stage of the filter, the ensemble of particles implicitly, nonparametrically describes the probability density $p(\underline{z})$, clearly not relying on a single mean and covariance.

4.4 Dynamic Sampling

Section 3.3 examined random prior or posterior sampling in the static context. Now suppose that we have a dynamic process $\underline{z}(t)$, for which we would like to generate random samples.

We continue to assume that the dynamic process $\underline{z}(t)$ obeys the first-order Gauss–Markov model of (4.1):

$$\underline{z}(t+1) = A(t)\underline{z}(t) + B(t)\underline{w}(t), \quad \underline{z}(0) \sim \mathcal{N}(\underline{z}_o, P_o), \tag{4.160}$$

where $\underline{w}_i$ is a white Gaussian noise process with identity covariance. The first-order Gauss–Markovianity is crucial, because it allows the statistics of the next time step to be expressed in terms of the current state:

$$\underline{z}(t+1)|\underline{z}(t) \sim \big(A(t)\underline{z}(t), B(t)B^T(t)\big). \tag{4.161}$$

Furthermore, the matrix $B(t)$ is already the square root of $B(t)B^T(t)$, the covariance of the added noise. Therefore prior sampling proceeds recursively, such that

$$\underline{z}(t+1) = A(t)\underline{z}(t) + B(t)\underline{w}(t) \tag{4.162}$$

and only the initial covariance matrix $P_o = \Gamma_o\Gamma_o^T$ needs to be decomposed per (2.76), where we initialize the recursion with

$$\underline{z}(0) = \underline{z}_o + \Gamma_o\underline{w}. \tag{4.163}$$

However, it should be noted that in a derived dynamic model the parameters B_i are not, in fact, normally known; rather the inferred dynamic process takes the form

$$\underline{z}(t+1) = A(t)\underline{z}(t) + \underline{q}(t), \quad \underline{z}_o \sim \mathcal{N}(\underline{z}_o, P_o), \quad \underline{\bar{w}}(t) \sim \mathcal{N}\big(\underline{0}, W(t)\big) \tag{4.164}$$

where $\underline{\bar{w}}(t)$ is a noise process with covariance $W(t)$. In this case we must decompose $P_o = \Gamma_o\Gamma_o^T$ as before, but also $W(t) = \Gamma_t\Gamma_t^T$ for each time step, a great deal more work than before. In this case the key to efficiency is somehow to keep the size of matrix $W(t)$, and thus the state dimension of $\underline{z}(t)$, as small as possible.

The problem of dynamic posterior sampling is discussed in Section 11.1 in the chapter on sampling.

4.5 Dynamic Estimation for Discrete-State Systems

Most of this text is concerned with the estimation of continuous-state systems, however there are contexts (particularly in Chapter 7) where the dynamic state is discrete. We begin by characterizing the discrete-state problem, followed by the Viterbi algorithm, recursive like the Kalman filter, but for discrete-state systems.

4.5.1 Markov Chains

We have extensively looked at the first-order Gauss–Markov dynamic model of (4.1). The first-order discrete-state analogue of (4.1) is known as a Markov chain [42, 137]. Given the discrete state $z(t) \in \Psi_z$, where $\Psi_z = \{\psi_i\}$ is the set of possible state values, we have a discrete set of state-to-state transition probabilities

$$a_{ij} = \Pr\big(z(t+1) = \psi_i | z(t) = \psi_j\big). \tag{4.165}$$

Thus the dynamic matrix A completely characterizes the transition statistics from t to $t+1$, and indeed for all time if the process is time-stationary.

It is possible to have higher-order processes, such as

$$\Pr\big(z(t+1) = \psi_i | z(t) = \psi_j, z(t-1) = \psi_k\big), \tag{4.166}$$

or nonscalar processes, such as

$$\Pr\left(\begin{bmatrix} z_1(t+1) = v \\ z_2(t+1) = w \end{bmatrix} | \begin{bmatrix} z_1(t) = x \\ z_2(t) = y \end{bmatrix}\right). \tag{4.167}$$

Both of these cases can, in principle, be converted into the first-order scalar form using a state augmentation process very similar to that in Section 4.1.1, so our focus is on the first-order scalar case.

If we let $\underline{p}(t)$ represent the state probability distribution at time t,

$$\underline{p}(t) \triangleq \begin{bmatrix} \Pr(z(t) = \psi_0) \\ \Pr(z(t) = \psi_1) \\ \vdots \end{bmatrix}, \tag{4.168}$$

then the stochastic evolution of the process is elegantly written as

$$\underline{p}(t+1) = A\underline{p}(t). \tag{4.169}$$

What is attractive about this formulation is that a wide variety of statistical properties of the Markov chain, such as connectedness, periodic cycles, and steady-state behaviour, can be inferred from an analysis of A.

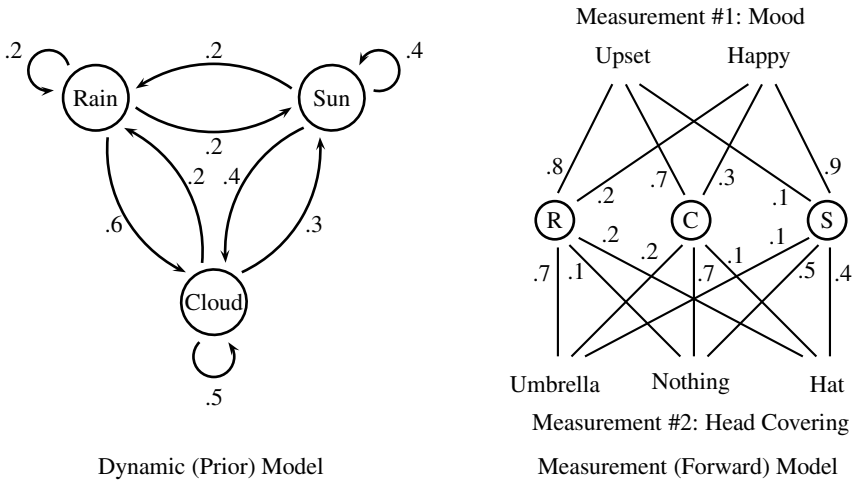


Fig. 4.4. A discrete-state first-order dynamic model. The dynamic model, left, is described in terms of state-transition probabilities. The measurement model, right, describes the state-dependent probabilities of two observations — mood and head-covering. Clearly the model is simplistic: people probably would be happy if it were to rain after a few weeks of sun, however such a model requires memory and is *not* first order.

Analogously to the continuous-state observation equation, here the discrete state $z(t)$ of the Markov chain influences some observation $\underline{m}(t)$, such that the forward model is expressed as a conditional likelihood

$$c(\underline{m}, \psi) = p\big(\underline{m}(t) | z(t) = \psi\big) \tag{4.170}$$

if the measurements are continuous, or as

$$c_{ij} = \Pr\big(m(t) = m_i \in \Psi_m | z(t) = \psi_j\big) \tag{4.171}$$

if the measurement at each time is a discrete scalar.

An example of a discrete-state dynamic model and a corresponding measurement model are shown in Figure 4.4.

4.5.2 The Viterbi Algorithm

We are given a scalar Markov chain, characterized by A, together with observations $m(t), 1 \leq t \leq \tau$. A state sequence

$$\underline{z}(\tau) = \begin{bmatrix} z_1 \\ \vdots \\ z_\tau \end{bmatrix} \in \Psi^\tau \tag{4.172}$$

is composed of states over time, such that there are $|\Psi^\tau| = |\Psi|^\tau$ possible sequences. We wish to estimate such a state sequence, in particular the MAP estimate

$$\hat{\underline{z}}_{\text{MAP}} \triangleq \arg_{\underline{z}\in\Psi^\tau} \max \Pr\big(\underline{z}|m(1),\ldots,m(\tau)\big). \tag{4.173}$$

Although the total number $|\Psi|^\tau$ of possible state sequences is impossibly large, because $\underline{z}$ is first-order Markov it is possible to use methods of induction or dynamic programming to find a recursive solution, the Viterbi algorithm [320], to (4.173).

Let

$$p_i(t|s) = \Pr\big(z(t) = \psi_i|m(1),\ldots,m(s)\big) \tag{4.174}$$

and suppose that $\underline{p}(1|0)$ is given, analogous to the Kalman filter initialization $\underline{z}_o, P_o$ at time $t = 0$. Then the prediction step is given by the Markov chain statistics (4.169):

$$\underline{p}(t|t-1) = A\underline{p}(t-1|t-1) \tag{4.175}$$

and the update step follows from Bayes' rule:

$$p_i(t|t) = p_i(t|t-1) \cdot \frac{\Pr\big(m(t)|z(t)=\psi_i\big)}{\Pr\big(m(t)\big)}. \tag{4.176}$$

The resulting vector $\underline{p}(t|t)$ is the probability of the possible values of $z(t)$, summing over all possible state sequences before time t.

Suppose we are given a state sequence and its associated probability:

$$\underline{z}(t-1) = \begin{bmatrix} z(1) \\ \vdots \\ z(t-1) \end{bmatrix} \qquad \Pr\big(\underline{z}(t-1)|t-1\big). \tag{4.177}$$

Then the predicted probability of the next state transition is

$$\Pr\big(\big[\underline{z}(t-1),\, z(t)\big]\,|\,t-1\big) = a_{z(t),z(t-1)} \Pr\big(\underline{z}(t-1)|t-1\big) \tag{4.178}$$

therefore, as in (4.176), the updated probability is

$$\Pr\big(\big[\underline{z}(t-1),\, z(t)\big]\,|\,t\big) = a_{z(t),z(t-1)} \Pr\big(\underline{z}(t-1)|t-1\big) \frac{\Pr\big(m(t)|z(t)\big)}{\Pr\big(m(t)\big)}, \tag{4.179}$$

where the calculation of $\Pr\big(m(t)\big)$ may be difficult, and has no effect on the relative state-sequence probabilities, so that the normalization factor in the denominator can be ignored.

Not all state sequences are very likely, however. It is the MAP estimate $\hat{\underline{z}}_{\text{MAP}}$ for which we would like to derive a forward-recursion. Of all possible state sequences

which end up in state ψ_i at time $t-1$, let $\underline{\hat{z}}^i_{\text{MAP}}(t-1|t-1)$ be the most likely one. Then, using (4.179), the MAP estimate at the next time step is just the most likely of all possible sequences:

$$\underline{\hat{z}}^j_{\text{MAP}}(t|t) = \left[\underline{\hat{z}}^i_{\text{MAP}}(t-1|t-1),\, \psi_j\right] \quad \text{where} \quad i = \arg_k \max \Pr\Big(\left[\underline{\hat{z}}^k_{\text{MAP}}(t-1|t-1),\, \psi_j\right] \,|\, t\Big). \tag{4.180}$$

At the final time step, the MAP estimate is given by that sequence with the highest probability:

$$\underline{\hat{z}}_{\text{MAP}}(\tau|\tau) = \max_i \left\{\underline{\hat{z}}^i_{\text{MAP}}(\tau|\tau)\right\}. \tag{4.181}$$

Figure 4.5 provides a visualization of the Viterbi estimation process for the model of Figure 4.4. Although the three optimal paths $\underline{\hat{z}}^i_{\text{MAP}}(t|t)$ happen to be coincident, this is not a requirement; it is perfectly possible to have the $|\Psi|$ state-sequences with different state histories.

4.5.3 Comparison to Kalman Filter

The Viterbi and Kalman filters both solve a discrete-time estimation problem and are both based on a prediction-update framework, however there are significant differences.

First, the Kalman filter is optimum in a least-squares sense, whereas the Viterbi method finds the globally optimum MAP solution. For a discrete-state problem, the estimate is either correct or incorrect, so a least-squares notion of "closeness" is inappropriate.

Second, it is important to realize that the Viterbi method is essentially smoothing, estimating the optimum state sequence given *all* of the measurements, in contrast to the Kalman filter. The distinction arises because the discrete-state system allows the enumeration of all $|\Psi|$ possible state values at some point in time, and the first-order nature of $z(t)$ implies that it is possible to find a recursive estimator, dependent only on the $|\Psi|$ optimal solutions up to time t. The number of solutions to preserve increases rapidly with model order, since an nth-order estimator is conditioned on the value of n successive states, requiring $|\Psi|^n$ state sequences.

Finally, the Viterbi method maintains $|\Psi|$ different state sequences, qualitatively similar to the multiple-hypothesis approaches of the ensemble, unscented, and particle Kalman filters of Section 4.3.4. However for the discrete-state case, the $|\Psi|$ state sequences enumerate *all possible* optimal paths for a first-order system, whereas the multiple Kalman-like filters in Section 4.3.4 cannot possibly enumerate all paths, but increase the probability of a good estimate by improving robustness and better characterizing the estimator statistics in the presence of nonlinearities.

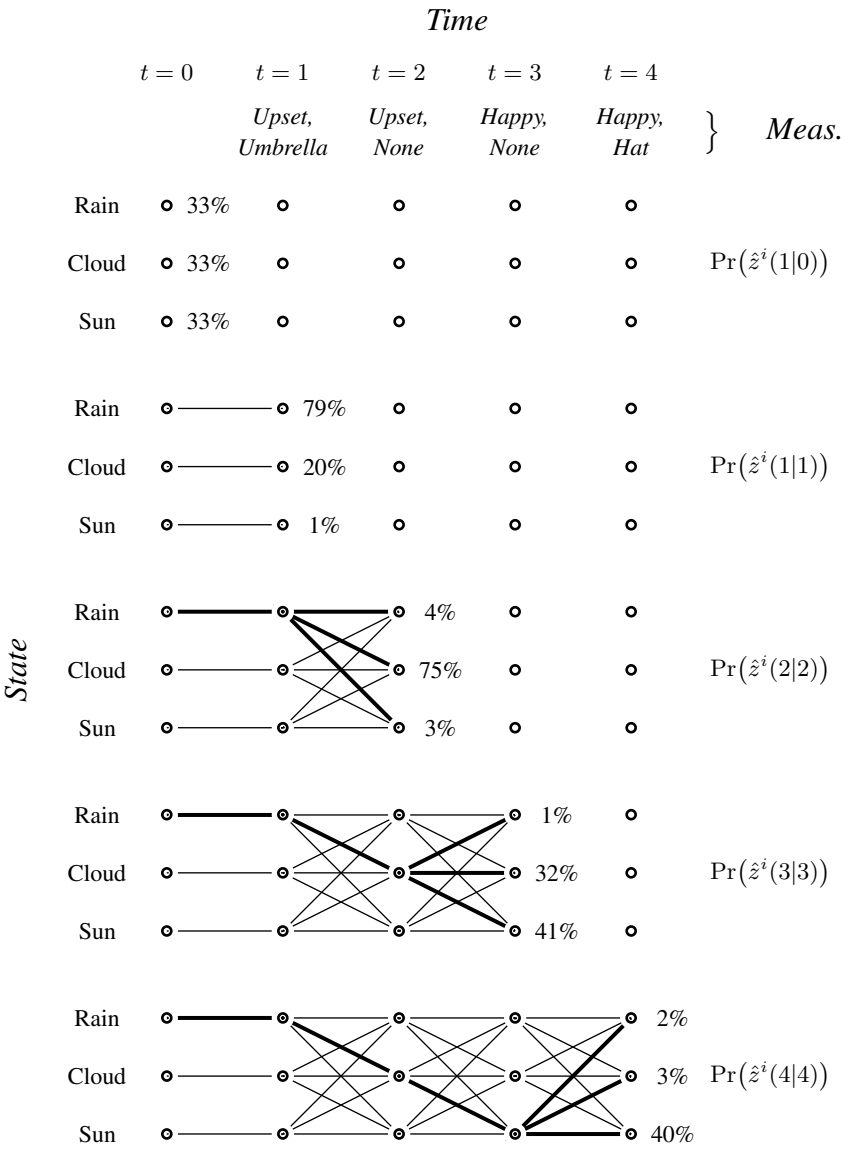


Fig. 4.5. The Viterbi algorithm, applied to the model of Figure 4.4. The model is initialized, followed by measurements over four time steps. Light lines show hypothetical state transitions, whereas the bold lines and probabilities correspond to the three state sequences preserved at each time step in $\{\hat{\underline{z}}^i_{\text{MAP}}(t|t)\}$.

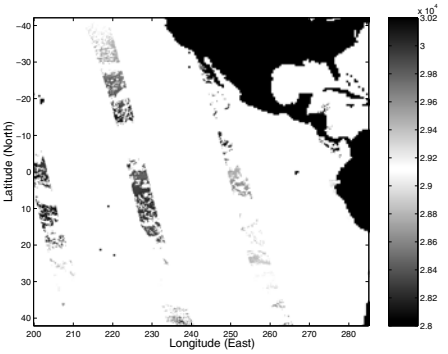


Fig. 4.6. Satellite observations of the ocean surface temperature for the month of October 1992: the measurements are accurate, in the sense of low-noise, however they are spatially located in narrow, sparse strips.

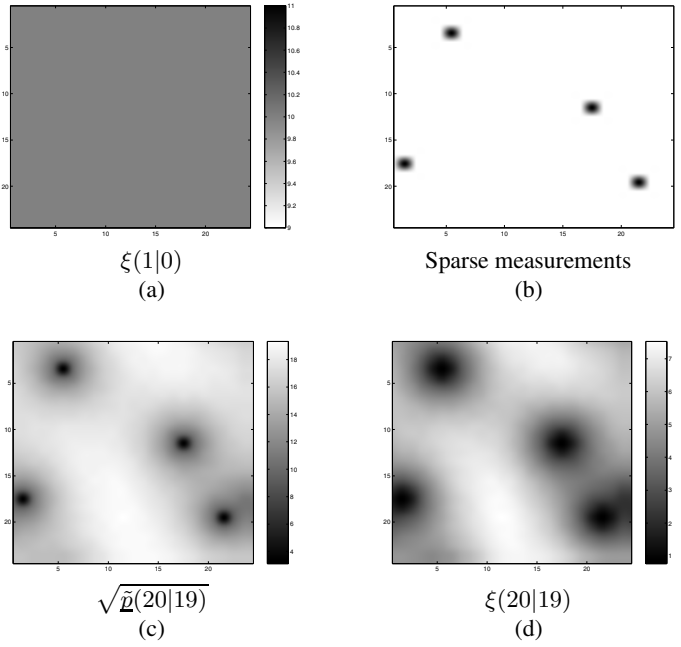


Fig. 4.7. Prediction: (a) The initial stationary correlation length of the prior model, which is updated with the four point measurements (b). After twenty prediction steps, (c) the error standard deviations and (d) the estimation error correlation length $\xi(20|19)$ are both clearly nonstationary.

Application 4: Temporal Interpolation of Ocean Temperature [191]

Consider the data shown in Figure 4.6: the ATSR satellite offers accurate observations of the ocean surface, however the orbital pattern of the satellite, the narrow view, and the interference by clouds end up giving us measurements in sparse strips, certainly nothing like a map.

What we want to do is to produce estimates *over time*, so that as measurements arrive we slowly fill in a dense map. Of course, the ocean surface is dynamic, so we would like something akin to a Kalman filter that takes the dynamics into account and produces dense estimates. A direct implementation of the Kalman filter is not possible, because the 512×512 size of the image is far too large, as it leads to a state vector with 262 144 elements, and correspondingly huge covariance matrices.

In principle the prediction step can be implemented along the lines of the extended Kalman filter: have some sort of dynamic model which includes ocean-surface currents etc, and use these dynamics to predict the estimates.

It is the update step which causes us problems. If the prior model $P(t|t-1)$ at some time step t is spatially stationary, then there are some tricks that we can use to implement the update step efficiently; here the multiscale method of Section 10.3 was used to solve the update step. However, as illustrated in Figure 4.7, if we have sparse measurements then the posterior $P(t|t)$ becomes *non*stationary, thus giving us a nonstationary prior for which effective approximate modelling is much more difficult, at time $t+1$.

To solve this problem [191], the nonstationary spatial prior was approximated as a weighted mix of stationary models, as described in Section 5.6.2. The results, shown in Figure 4.8 show the estimates filling in the dense map over time, as more and more measurements arrive.

Summary

Given a dynamic Bayesian estimation problem

$$\underline{z}(t+1) = A\underline{z}(t) + B\underline{w}(t) \qquad \underline{w}(t) \sim \mathcal{N}\big(\underline{0}, I\big) \tag{4.182}$$

$$\underline{m}(t) = C\underline{z}(t) + \underline{v}(t) \qquad \underline{v}(t) \sim \mathcal{N}\big(\underline{0}, R\big) \tag{4.183}$$

the optimum least-squares recursive estimator, the Kalman filter, can be formulated as

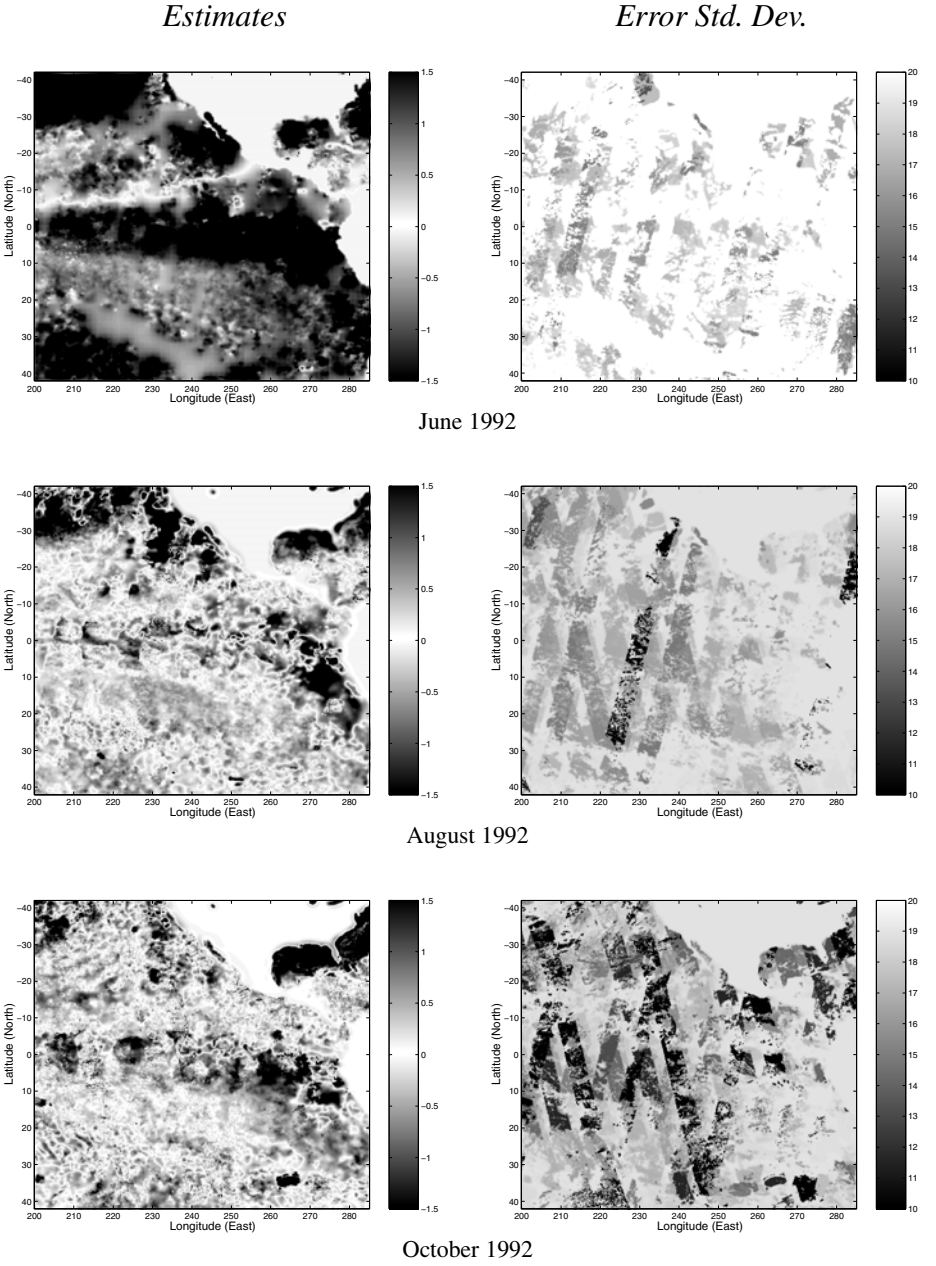


Fig. 4.8. Anomaly estimates and error standard deviations for the proposed estimator, applied to five months of ATSR sea-surface temperature data, starting in June, 1992.

Prediction:
$$\hat{\underline{z}}(t|t-1) = A\hat{\underline{z}}(t-1|t-1) \tag{4.184}$$
$$P(t|t-1) = AP(t-1|t-1)A^T + BB^T \tag{4.185}$$

Update:
$$\hat{\underline{z}}(t|t) = \hat{\underline{z}}(t|t-1) + K(t)\big(\underline{m}(t) - C\hat{\underline{z}}(t|t-1)\big) \tag{4.186}$$
$$P(t|t) = P(t|t-1) - K(t) \cdot C \cdot P(t|t-1) \tag{4.187}$$
$$K(t) = P(t|t-1) \cdot C^T \cdot \big(CP(t|t-1)C^T + R\big)^{-1}. \tag{4.188}$$

If matrices A, B, C, R are not a function of time, then we have the steady-state Kalman filter, in which only (4.184) and (4.186) are iterated.

The smoothed estimates $\hat{\underline{z}}(t|\tau), P(t|\tau)$ can be computed recursively using the Kalman smoother.

For Further Study

A great many books have been written on Kalman filtering, of which the textbooks by Anderson and Moore [8], Grewal and Andrews [151], Hayes [156], and Mendel [231] are a representative sample.

There is a Kalman filter toolbox available for MATLAB, and many open-source implementations can be found.

Sample Problems

Problem 4.1: No Measurements

In the regular Kalman filter, if at some time t we have *no* measurements, then presumably the update step should have no effect. Prove that this is so. That is, if there are no measurements, show that the regular Kalman update step reduces to

$$\hat{\underline{z}}(t|t) = \hat{\underline{z}}(t|t-1) \qquad P(t|t) = P(t|t-1), \tag{4.189}$$

meaning that we can skip the update step entirely at those iterations in which we have no measurements.

Problem 4.2: Update–Update Form

In the regular Kalman filter, substitute the prediction equations into the update equations in order to derive the update–update form of (4.82), (4.83).

Problem 4.3: Noncausal Filtering

It is straightforward to implement a noncausal Kalman filter by augmenting the state, as in (4.143), such that $\underline{z}(t)$ is estimated based on measurements up to time $t+\tau$, where τ is the degree of state augmentation in (4.143), (4.144).

The reason for doing such estimation is, of course, because it is likely that future measurements are relevant to estimating the present state. That is, the estimation error covariance $P(t|t+\tau)$ should decrease with increasing τ.

(a) Prove that $P(t|t+\tau)$ cannot increase with τ. That is, prove that

$$P(t|t+\tau+1) \leq P(t|t+\tau) \quad \text{for all } \tau \geq 0. \tag{4.190}$$

(b) Under what conditions of the dynamic model $A(t), B(t)$ is (4.190) met with equality, such that the error covariance does *not* decrease with τ?
(c) Under what conditions of the measurement model $C(t), R(t)$ is (4.190) met with equality, such that the error covariance does *not* decrease with τ?

Problem 4.4: Filter Implementation

Suppose we have the dynamic system

$$z(0) \sim \mathcal{N}(0,4) \qquad z(t) = \alpha z(t-1) + w(t) \qquad w(t) \sim \mathcal{N}(0,1) \tag{4.191}$$

and measurements

$$m(t) = z(t) + v(t) \qquad v(t) \sim \mathcal{N}(0,4) \tag{4.192}$$

We can use the dynamics (4.191) to synthesize a process $z(t)$ and then create measurements via (4.192), which can then be used by the Kalman filter to generate estimates $\hat{z}(t|t)$. Iterate over twenty time steps $0 \leq t \leq 20$:

(a) Set $\alpha = 0.9$. Plot $z(t), m(t), \hat{z}(t|t), \sqrt{P(t|t)}$.
(b) Set $\alpha = 2.0$. Plot $z(t), m(t), \hat{z}(t|t), \sqrt{P(t|t)}$.

Note that for $\alpha = 2$ the dynamic system is unstable, however does the Kalman filter still do a good job in estimating $z(t)$?

Our dynamic system is time-invariant, therefore the steady-state Kalman filter can be used. We assume that the Kalman filter is essentially in steady-state after twenty iterations, that is, that

$$P(20|20) \simeq P(\infty|\infty). \tag{4.193}$$

(c) Plot $P(20|20)$ as a function of α for $-3 \leq \alpha \leq 3$.

There is a great deal to observe and explain in this plot. For each of the following, give a short mathematical (equation-based) argument, but *also* a descriptive, insightful explanation.

(d) Why is the plot symmetric about $\alpha = 0$?
(e) Explain the value of $P(20|20)$ at $\alpha = 0$.
(f) Why do we see asymptotic behaviour in P for large $|\alpha|$?
(g) Explain the value of $P(20|20)$ at $\alpha = \infty$.

Part II

Modelling of Random Fields

5

Multidimensional Modelling

In principle, the extension to multidimensional $\underline{z}$ of the concepts of regularization from Chapter 2 and the static and dynamic estimators of Chapters 3 and 4 is perfectly straightforward. The only inherent limitation in the developed estimators is that the unknowns $\underline{z}$ and measurements $\underline{m}$ are required to be column vectors. Attempting to substitute matrices (e.g., a measured image) for $\underline{m}$ will yield meaningless results.

It is easy, however, to reorder the image Z on a two-dimensional grid Ω into a vector via *lexicographic* reordering:

$$Z = [\underline{z}]_{3\times 3} \triangleq \begin{array}{|c|c|c|}\hline z_{11} & z_{12} & z_{13} \\ \hline z_{21} & z_{22} & z_{23} \\ \hline z_{31} & z_{32} & z_{33} \\ \hline \end{array} \quad \begin{array}{c}\text{Lexicograph.} \\ \rightleftharpoons \\ \text{Unlexicograph.}\end{array} \quad \underline{z} = Z_{:} \triangleq \begin{bmatrix} z_{11} \\ z_{21} \\ \vdots \\ z_{33} \end{bmatrix} \tag{5.1}$$

There is flexibility, of course, in the specifics of how the image Z should be scanned into the vector; that is, whether by columns, as in (5.1), by rows, or in some other fashion. Unless specified otherwise, by default we will choose to stack by columns.

Clearly the above approach extends trivially to higher dimensions, first stacking a three-dimensional cube by planes, and next the stacked plane by columns, however a certain degree of nervousness regarding the size of the resulting vectors is appropriate.

All of the previously-developed ideas of conditioning, regularization etc. apply, except that the matrices involved (e.g., covariance P or constraints L) possess more complicated structures due to the rearranged nature of the lexicographical stacking. Nevertheless, working with two- and higher-dimensional spaces introduces difficulties which we did not consider in previous chapters. This chapter begins with a discussion of the challenges posed by higher-dimensional problems, followed by a discussion of multidimensional deterministic and statistical models.

P. Fieguth, *Statistical Image Processing and Multidimensional Modeling*, Information Science and Statistics, DOI 10.1007/978-1-4419-7294-1_5, © Springer Science+Business Media, LLC 2011

5.1 Challenges

Ultimately, *all* of the challenges related to multidimensional data processing reduce to a problem of size. In particular, for an $N \times N$ image Z, the associated vector $\underline{z} = Z;$ is of length N^2, thus the associated covariance matrix P_z is of size $N^2 \times N^2$. Even for a very modestly-sized image of 256×256 pixels, this corresponds to a covariance containing $2^{16} \cdot 2^{16} =$ four *billion* elements.

Given a large multidimensional problem, there are three operations which we need to be able to undertake, forming the primary challenges we need to consider:

STORAGE: We need to store matrices (on disk or in memory).

COMPUTATION: We need to compute matrix inverses, matrix–matrix products, and matrix–vector products.

MODELLING: We need to infer / specify certain model matrices (e.g., A, C, L, P, R).

To the extent that the above three steps are practical, corresponding to problems having up to roughly ten-thousand elements (that is, a 100×100 pixel image), the methods of Chapters 3 and 4 are immediately applicable. For much larger problems, such as the global sixth-degree 2160×1080 pixel oceanographic problem of Application 2, obviously much more care must be taken. In particular, the direct application of Chapter 3 to such a problem is computationally inconceivable, thus the problem either needs to be reformulated or made smaller, consequently leading to the following four challenges:

1. Dimensionality reduction: How to reduce the size of the problem, or to decompose it into multiple smaller pieces.
2. Efficient matrix storage: Methods for sparse matrix representation, to reduce storage requirements.
3. Efficient computation: Associated methods for efficient computation with sparse matrices.
4. Statistical modelling: How to model transformed problems, whether of reduced size, decomposed, sparse, or some other simplified or efficient representation.

The remainder of this chapter surveys methods to address these challenges, with specific approaches developed in greater detail in the following chapters.

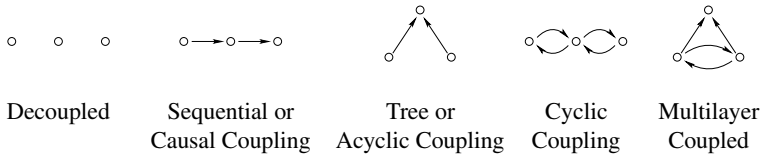


Fig. 5.1. The way in which state elements interact or are coupled strongly influences the difficulty of the problem. Five examples are shown here, in increasing order of difficulty.

5.2 Coupling and Dimensionality Reduction

The presence or absence of coupling between state variables plays a major role in the algorithmic complexity — the difficulty in formulating a solution — and associated computational complexity and numerical conditioning.

Figure 5.1 shows five examples, in increasing order of difficulty:

- In a *decoupled* problem the state elements do not interact, therefore there is no coupling to encode (reduced representation complexity) and each element can be modelled and processed independently.
- In a *sequential* problem, the elements do interact, however the interaction is causal, meaning that the elements can be *ordered* from start to end. Some recursive approach, such as a variant of a Kalman filter, may be applied to such problems.
- In *trees* or acyclic graphs (relationships having no loops) the behaviour is complicated by the less regular structure, and by the fact that multiple elements (the leaves of the tree) can influence one state (the root). Nevertheless the absence of loops allows generalizations of the Kalman filter to be applied to such problems, usually at very low computational complexity.
- In cyclically *coupled* (loopy) problems, the bidirectionality of influence makes the problem *much* harder than the decoupled and causal cases. Many algorithmic solutions to such problems are iterative, updating first one state, then the other, then back to the first and so on.
- Coupled *multilayer* problems are like coupled ones, but their heterogeneous form makes it more difficult to transform them into something simpler, whereas it may be possible to convert a regularly-structured coupled problem.

With the exception of problems with unusually convenient structures (see, in particular, Section 8.3), in approaching a multidimensional problem of any significant size

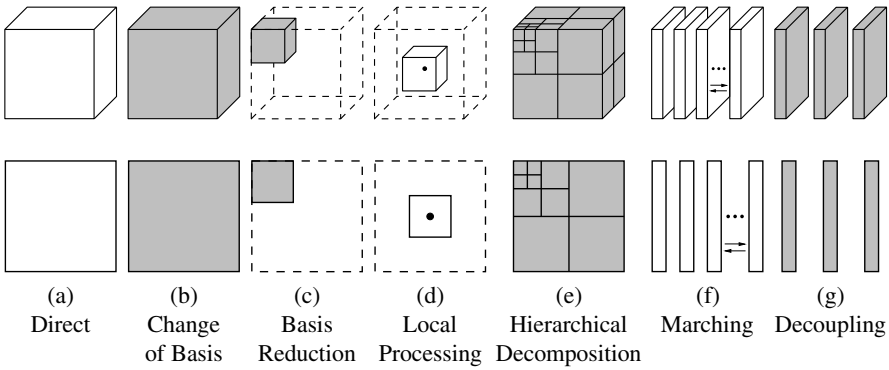


Fig. 5.2. A wide variety of approaches exists for addressing large multidimensional problems. Above are sketched a quick survey of seven of the broadest approaches to 3D (top) and 2D (bottom) problems: direct, change of basis, reduction of basis, local processing, hierarchical processing, marching, and decoupling. Grey regions show changed or transformed coefficients; solid lines within dashed regions show a reduction in the number of coefficients.

our goal is frequently one of simplifying the coupling attributes, specifically breaking loops via some sort of decomposition or transformation, so that a looped graph becomes acyclic, whether tree-based, sequential, or decoupled.

Although the details are left for other chapters to develop further, Figure 5.2 gives a quick overview of the most promising categories:

(a) Direct (Chapter 3): The problem is set up in dense form, and solved directly per the usual equations. This approach is infeasible for all but the most trivially sized problems.

(b) Change of Basis (Section 8.1): Although the size of the problem is not reduced, by reducing the correlation among state elements the overall conditioning of the problem is improved.

(c) Reduction of Basis (Section 8.2): An extension of change of basis, in many contexts the redundancy among the state elements implies that the original problem can be very accurately represented using a reduced state, essentially no different from the principle of image compression.

(d) Local Methods (Section 8.2.3): A special case of reduction of basis, here each estimate is based only on a local pruned version of the original problem. Except for those problems with localized correlations, this approach is normally rather poor.

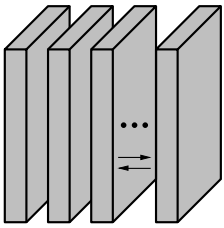


Figure 5.2(b,f)
Basis Change +
Marching

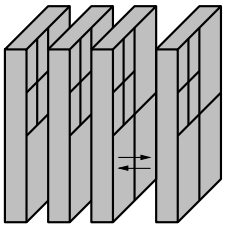


Figure 5.2(e,f)
Hierarchical Basis +
Marching

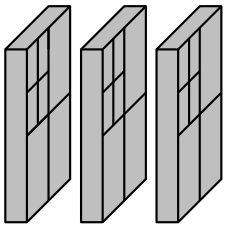


Figure 5.2(c,e,g)
Basis Reduction +
Decoupling + Hierarchy

Fig. 5.3. Three examples of hybrid approaches to three-dimensional modelling, based on combining two or more methods from Figure 5.2: (b,f) applying a global change of basis and processing plane by plane using a marching approach, (e,f) applying a hierarchical approach to each of the planes in a marching approach; (c,e,g) decomposing the problem into independent planes, reducing the basis by eliminating most of the insignificant planes, and applying a hierarchical approach to each plane independently.

(e) Hierarchical or Scale Recursion (Figure 5.5): A specialized change of basis, such that the elements can be organized into a hierarchy of scales.

(f) Marching (Section 10.1): Applying a Kalman filter to one axis of a multidimensional problem, solving the problem iteratively over columns or planes.

(g) Decoupling (Section 8.2.1): Apply an optimal change of basis, decoupling the problem into independent pieces. Although the optimal change of basis is too difficult to apply to a whole d-dimensional problem, if the problem is spatially stationary then the optimal one-dimensional decoupling may be found along some dimension, leaving an ensemble of simpler $(d-1)$ dimensional problems.

Clearly more than one approach may be used in a given context; three illustrations of such hybrid schemes are shown in Figure 5.3.

The topic of hierarchical methods deserves some additional comment. There are, typically, two classes of motivations for a hierarchical method:

PRAGMATICS: Hierarchical methods, particularly hierarchical changes of basis (such as wavelets) are often found to be very effective at improving problem conditioning, and are thus chosen for their numerical properties.

PHILOSOPHY: For most large-scale multidimensional problems, there is some recognition [110, 148, 273] that the fine-scale details in one part of the problem are unnecessary in modelling some distant part of the problem, as illustrated in Fig-

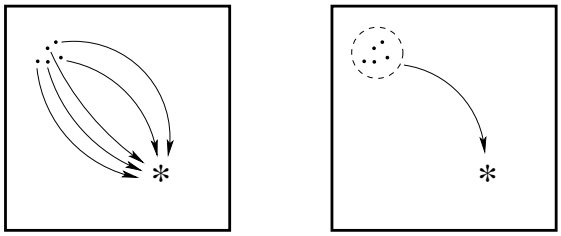


Fig. 5.4. (a) Knowing the details of individual, distant points is probably not necessary in estimating the value of a random field at point *. Instead, (b) a hierarchical approach is motivated whereby the details are aggregated into a much simpler form [148, 273].

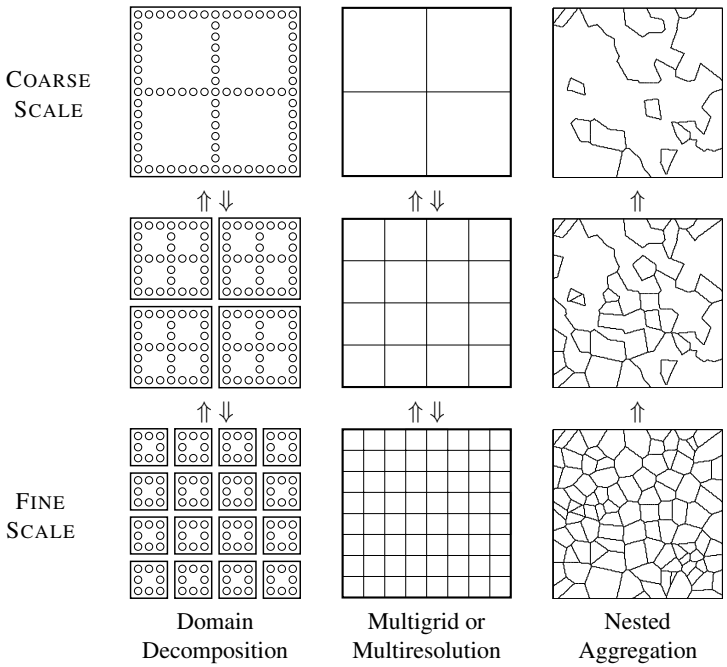


Fig. 5.5. Three basic approaches to hierarchical modelling. Domain decomposition partitions the domain into small pieces, multiresolution methods represent a domain at coarse scales using relatively few coefficients, and nested aggregation groups small regions to produce fewer but larger ones.

ure 5.4. Instead, it may be preferable to aggregate these details (and then to repeatedly aggregate the aggregations, leading to a hierarchy).

Very generally, three classes of hierarchical methods exist, illustrated in Figure 5.5, all of which will be developed in further detail as appropriate:

1. Domain decomposition: Also known as "divide-and-conquer," a large problem is repeatedly divided into some number of smaller decoupled subproblems. At each scale, only the boundaries of the decoupled subproblems are of interest, the interior of each subproblem having been solved at a finer scale. Nested dissection (Section 9.1.3) and statistical multiscale (Section 10.3) are examples of this approach.

2. Multiresolution: The problem is represented at a variety of resolutions, such that the coarse scales have only few state values representing low-resolution aspects of the random field, with progressively finer details at finer scales. Although superficially similar to domain decomposition, there are two significant differences:

   (a) A multiresolution approach normally does *not* decouple into multiple problems at finer scales. There is one problem to solve at each scale.

   (b) Methods of domain decomposition normally do not represent a random field at varying *resolutions*. Although one talks about "coarse" and "fine" scales, the representation at the coarsest scale is normally a *subset* of the finest scale, not a *low-resolution* version of the finest scale.

   Hierarchical basis change (Sections 8.4.1 and 8.4.2) and the multigrid algorithm (Section 9.2.5) are examples of this approach.

3. Nested aggregation: In both the decomposition and multiresolution approaches the geometry of the coarsest scale — that is, the extent of the coarse scale elements on the finest scale — is predetermined. However, if a random field consists of irregularly-shaped patches, it may be inappropriate to represent this random field using large, square pixels at a coarse scale. In region aggregation, the coarser scales are constructed by the repeated iterative aggregation of finer scales, leading to arbitrarily-shaped regions whose shapes are driven by the finest-scale observations.

5.3 Sparse Storage and Computation

5.3.1 Sparse Matrices

In the event that the covariance P for an $N \times N$ image Z is completely nonstationary without any structure or regularity, there is not really any alternative to specifying the

complete, dense $N^2 \times N^2$ matrix P. On the other hand, there is something that feels very wrong about needing to specify and store $N^4/2$ parameters[1] for an image of N^2 pixels.

Indeed, the solution in all cases is to assert, assume, or derive a model which leads to *sparse* matrices. In some cases this sparsity is obvious, such as with diagonal matrices. For example, very frequently it is reasonable to assume that measurements have uncorrelated errors, in which case R is diagonal, and so only the vector

$$\underline{r}_d^T = \left[r_{11} \dots r_{nn}\right] = \text{diag}(R) \tag{5.2}$$

needs to be stored. As a limiting case, if the measurement noise is uncorrelated and also stationary, then $R = \sigma^2 I$ and the entire matrix is represented by the single scalar variance σ^2.

Although there is a large literature of sparse matrix techniques [141], many of which apply to arbitrary patterns of matrix sparsity, in general the regular pixellated nature of images and volumes leads to fairly regular structures. Figure 5.6 shows two such examples for 10×10 pixel images Z. Suppose that any pixel $z_{i,j}$ is correlated only with some subset of its immediate neighbours; then the corresponding covariance[2] P is a *diagonally banded* matrix, where the bands are the consequence of the column-wise lexicographic reordering of Z. In these examples, the thickness of the main diagonal band (3 and 5, respectively) is determined by the number of *vertically*-related elements, whereas the number of band sets (also 3 and 5) is determined by the number of *horizontally*-related elements.

Thus in Figure 5.6(a), rather than using some complicated sparse representation for the covariance, it is much simpler and more efficient to recognize the banded structure of P, representing the b diagonal bands of P as a dense $b \times n$ matrix.

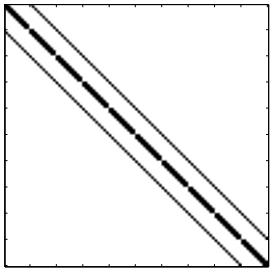
Although sparse banded matrices sound very straightforward, the challenge to sparse modelling stems from the following:

- The product of sparse matrices is normally more dense than the original, and
- The inverse of a sparse matrix is normally dense.

That is, the sparse structure of a given model tends to disappear very quickly. Much of the essence of the methods described in this book is the preservation, whether implicitly or explicitly, of model sparseness.

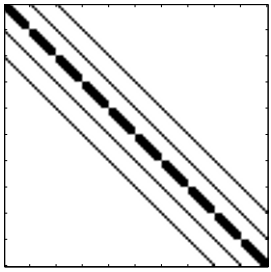
[1] The symmetry of a covariance matrix means that there are actually $q(q+1)/2$ parameters to specify for a $q \times q$ matrix.

[2] This example is for illustration purposes only. Most banded matrices are not positive-definite, and would therefore not be a valid covariance. In practice, it is covariance *inverses* which are banded, not the covariances themselves.



		$z_{-1,0}$		
	$z_{0,-1}$	$z_{0,0}$	$z_{0,+1}$	
		$z_{+1,0}$		

(a) 5-Banded



		$z_{-2,0}$		
		$z_{-1,0}$		
$z_{0,-2}$	$z_{0,-1}$	$z_{0,0}$	$z_{0,+1}$	$z_{0,+2}$
		$z_{+1,0}$		
		$z_{+2,0}$		

(b) 9-Banded

Fig. 5.6. Two illustrations of sparsity, both corresponding to a 10×10 two-dimensional domain. The lower arrays show a lexicographic wrapping of one row of the top matrices, showing the (a) five or (b) nine elements whose interactions are being modelled. The explanation of the patterned thick middle bands is complex, to be discussed in Section 5.5, and is due to the choice of boundary conditions at the image boundary.

5.3.2 Matrix Kernels

In seeking to model large multidimensional domains, an extremely important special case is that of spatial stationarity, where the spatial statistics or the form of a dynamic model do not vary from one state element to the next. In such a case, defining the nature of the matrix at some location implicitly defines the matrix at all other locations; such an implicit representation is referred to as a matrix *kernel*, and is far more sparse than any banded representation.

Consider two-dimensional random field $Z = \{z_{i,j}\}$, where the lexicographically reordered field has a given covariance

$$\underline{z} = [Z]_{:} \sim P. \tag{5.3}$$

If Z is spatially stationary then, ignoring the boundaries of the domain for now, the statistics are shift invariant:

$$E[z_{a,b}z_{i,j}] = E[z_{a+x,b+y}z_{i+x,j+y}] = E[z_{0,0}z_{i-a,j-b}] \tag{5.4}$$

from which it is clear that *all* of the field correlations can be inferred from the correlation of one element $z_{0,0}$ with the rest of the random field Z. That is, the covariance

kernel $\mathcal{P}$ is

$$\mathcal{P} \triangleq E[z_{0,0}Z]. \tag{5.5}$$

If $z_{0,0}$ is the first element of $[Z]_:$ and $\underline{p}_1$ the first column of covariance P, then

$$\underline{p}_1 \equiv E\left[z_{0,0}\,[Z]_:\right] \tag{5.6}$$

and the kernel may be succinctly represented as

$$[\mathcal{P}]_: = \underline{p}_1 \tag{5.7}$$

and we see that the kernel is nothing more than the reordering of the first column of the covariance. In the unusual case that the correlations are zero beyond some spatial offset q, then the kernel can be shrunk to represent only the local, nonzero correlations:

$$\mathcal{P} = E\left\{ z_{0,0} \cdot \begin{bmatrix} z_{-q,-q} & \cdots & z_{0,-q} & \cdots & z_{q,-q} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ z_{-q,0} & \cdots & z_{0,0} & \cdots & z_{q,0} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ z_{-q,q} & \cdots & z_{0,q} & \cdots & z_{q,q} \end{bmatrix} \right\} \tag{5.8}$$

The kernel $\mathcal{P}$ contains everything needed to reconstruct any entry of covariance P:

$$E\left[z_{a,b}z_{i,j}\right] = \begin{cases} 0 & \text{if } |a-i| > q \text{ or } |b-j| > q \\ \mathcal{P}_{i-a,j-b} & \text{if } |a-i| \le q \text{ and } |b-j| \le q. \end{cases} \tag{5.9}$$

A second important class of matrix kernels corresponds to linear dynamic matrices. Suppose we consider dynamics of the form

$$\underline{z}(t+1) = A\underline{z}(t). \tag{5.10}$$

If the dynamics are spatially stationary,[3] then how A acts to determine $z_{a,b}(t+1)$ is just a space-shifted version of how A acts to determine $z_{a+1,b-2}(t+1)$. That is, we can define a set of weights $\alpha_{i,j}$ such that

$$z_{a,b}(t+1) = \sum_i \sum_j \alpha_{i,j} z_{a+i,b+j}(t). \tag{5.11}$$

We recognize this as a convolution; collecting the weights $\alpha_{i,j}$ into a kernel $\mathcal{A}$, we have a succinct version of (5.11) as

$$A\underline{z}(t) \equiv [\mathcal{A} * Z(t)]_:. \tag{5.12}$$

[3] As written, the dynamics in (5.10) are also *temporally* stationary, however this is not required. It is perfectly reasonable to have a spatially stationary time-varying kernel $\mathcal{A}(t)$.

The compactness and elegance of this kernel approach make it particularly useful in the development of deterministic (Section 5.5) and stochastic (Section 5.6) models, particularly because, in contrast to the dynamics matrix A or covariance P, the kernels $\mathcal{A}, \mathcal{P}$ have the same numbers of dimensions as Z and allow a fairly intuitive understanding of the implied dynamics or statistics. Ideally, we work entirely in the kernel domain, without lexicographical reordering at all, such as

$$Z(t+1) = \mathcal{A} * Z(t) \qquad \mathcal{P} = E\left[z_{\underline{0}} Z\right]. \tag{5.13}$$

The shorthand $\mathcal{A} * Z$ should not quite be taken literally; indeed, the convolution (Appendix C.1) $\mathcal{A} * Z$ produces a result on a larger domain than that of Z. Instead, the convolution is understood to apply over most of the domain, with certain exceptions at the boundaries.

To elaborate further on this point, it is important to realize that matrix kernels can be applicable in problems which are spatially nonstationary (as shown, for example, with boundary conditions in Section 5.5.1). Although arbitrary nonstationarities would require preserving the entire dynamics A or covariance P, if *most* of the field is stationary, characterized by a single kernel, with only rare exceptions (such as at the edges of the domain, or at region boundaries within the domain), then it is possible to characterize the nonstationary behaviour by a relatively small number of stationary kernels.

Strictly speaking, matrix kernels do not need to be rectangular in shape and only need to store nonzero elements. When we sketch a kernel it is often convenient to show only these nonzero elements, however in implementation it is usually most convenient, computationally, to store the kernel as a small, dense, rectangular array.

5.3.3 Computation

The dominant component of computational complexity invariably revolves around matrix multiplication, matrix inversion, and (rarely) matrix eigendecomposition or positive-definiteness testing. We summarize below the computational issues associated with dense, banded, and kernel matrix representations.

Dense Matrices

Given dense matrices $A \in \mathbb{R}^{n \times k}, B \in \mathbb{R}^{n \times k}, C \in \mathbb{R}^{k \times p}, D \in \mathbb{R}^{k \times k}$, then the following complexities are well known:

- Computing $A + B$ has complexity $\mathcal{O}(nk)$,
- Computing $A \odot B$ has complexity $\mathcal{O}(nk)$,

- Computing $A \cdot C$ has complexity $\mathcal{O}(nkp)$,
- Computing D^{-1} has complexity $\mathcal{O}(k^3)$.

It should be mentioned, in passing, that faster algorithms are known for both multiplication and inversion. The oft-cited Strassen's method [294] which computes a matrix inverse in $\mathcal{O}(k^{\log_2 7}) \simeq \mathcal{O}(k^{2.807})$, and faster methods to approximately $\mathcal{O}(k^{2.4})$ have been developed, all based on fast methods for matrix multiplication [247]. In general, however, the implementation complexity of these methods limits their use.

The computation of a matrix determinant is a more subtle question than matrix addition or multiplication. In particular, because the matrix determinant may be used to address questions of matrix singularity or positive-definiteness, it is important to establish whether a determinant is zero, positive, or negative. Therefore numerical rounding errors in determinant computation may be unacceptable, so there exists literature [1] on *exact* determinant computation for matrices having integer entries, with a complexity of approximately $\mathcal{O}(n^4 \ln n)$. In practice, the MATLAB evaluation of a matrix determinant for a nonsingular $n \times n$ matrix is approximately $\mathcal{O}(n^3)$.

Given an $n \times n$ matrix, testing its positive-definiteness involves the application of Sylvester's test (Appendix A.3), requiring the computation of determinants of 1×1, 2×2, $\ldots n \times n$ matrices, for a total complexity of $\mathcal{O}(n^4)$.

Banded Matrices

If matrices are sparse, then the computational issues are somewhat different. For a banded, sparse $n \times n$ matrix A, let

$$\mathcal{B}_A(i) \triangleq \begin{cases} 0 & \text{if, for all } j,\ a_{j+i,j} = 0 \\ 1 & \text{if, for at least one } j,\ a_{j+i,j} \neq 0 \end{cases} \qquad |i| < n. \tag{5.14}$$

That is, $\mathcal{B}_A(i)$ indicates whether diagonal band i is present in matrix A.

Then bands from either matrix contribute to the sum in matrix addition:

$$\mathcal{B}_{A+B} = \text{sign}(\mathcal{B}_A + \mathcal{B}_B), \tag{5.15}$$

and matrix multiplication corresponds to a convolution in band space:

$$\mathcal{B}_{AB} = \text{sign}(\mathcal{B}_A * \mathcal{B}_B), \tag{5.16}$$

a result that is easily proved:

$$\text{Given} \qquad C = A \cdot B \quad \Rightarrow \quad c_{ik} = \sum_j a_{ij} b_{jk} \tag{5.17}$$

from which it follows that

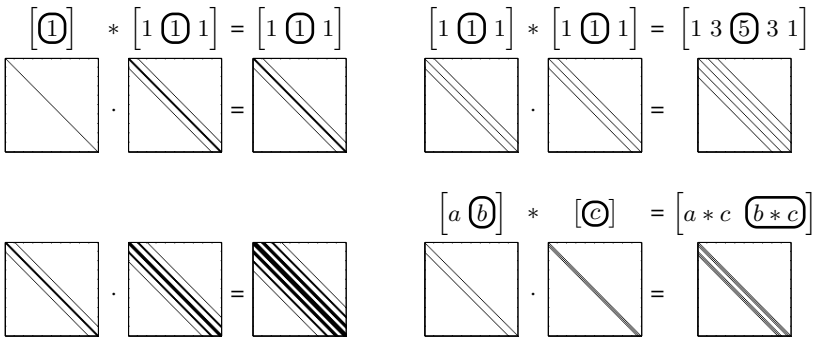


Fig. 5.7. The convolutional nature of sparse banded-matrix multiplication. Each matrix represents a lexicographically ordered two-dimensional problem; thick bands indicate interactions between vertically-adjacent elements, spaced bands indicate interactions between horizontally-adjacent elements. The circled elements indicate the kernel origin.

$$\mathcal{B}_C(i-k) = 1 \quad \text{if} \quad \sum_j \mathcal{B}_A(i-j)\mathcal{B}_B(j-k) > 0 \tag{5.18}$$

$$\mathcal{B}_C(i) = \operatorname{sign}\Big\{\sum_j \mathcal{B}_A(i-j)\mathcal{B}_B(j)\Big\} \tag{5.19}$$

$$= \operatorname{sign}\big\{\mathcal{B}_A(i) * \mathcal{B}_B(i)\big\}. \tag{5.20}$$

Four illustrations of this convolutional behaviour are shown in Figure 5.7. As each band in a banded matrix is associated with a single entry in its associated kernel (with the main diagonal corresponding to the kernel origin), that a convolution appears here is directly related to the multiplication of matrix kernels, discussed below.

Banded matrix inversion is somewhat more problematic. Whereas the product of two banded matrices yields another, slightly denser, banded matrix, the inverse of a banded matrix is typically *full*. It is possible, however, to formulate sparse, banded *approximations* to the matrix inverse [278]. A simple approach is to write a matrix $P = D + N$ in terms of its diagonal band D and nondiagonal bands N. If P is diagonally dominant, meaning that $D^{-1}N$ is stable, then there is a series approximation to P^{-1},

$$P^{-1} = D^{-1} - D^{-1}ND^{-1} + D^{-1}ND^{-1}ND^{-1} - \dots\,, \tag{5.21}$$

where the number of series terms depends on the degree of diagonal dominance. That this series is valid is easily seen by computing the product

$$\begin{aligned} P \cdot P^{-1} &= \big(D+N\big) \cdot \big(\; D^{-1} \;\; -D^{-1}ND^{-1} \;\; +D^{-1}ND^{-1}ND^{-1} \;\; -\dots\big) \\ &= \qquad\quad I \;\; -ND^{-1} \;\; +ND^{-1}ND^{-1} \;\; -\dots \\ &\qquad\quad +ND^{-1} \;\; -ND^{-1}ND^{-1} \;\; +\dots \end{aligned} \tag{5.22}$$

If N is b-banded, then the series builds up the dense matrix inverse by summing smaller and smaller terms having more and more bands:

$$P^{-1} \simeq \underbrace{D^{-1}}_{1 \text{ band}} - \underbrace{D^{-1}ND^{-1}}_{b^2 \text{ bands}} + \underbrace{D^{-1}ND^{-1}ND^{-1}}_{b^4 \text{ bands}} - \dots \tag{5.23}$$

Matrix Kernels

Basic matrix operations can be undertaken directly in the kernel domain (Section 5.3.2). As much as possible, all spatially stationary parts of a spatial statistical problem should be kernel-represented.

If we take the sum of two matrices then

$$(A+B)\underline{x} = A\underline{x} + B\underline{x} = [\mathcal{A} * X]_{:} + [\mathcal{B} * X]_{:} = [(\mathcal{A}+\mathcal{B}) * X]_{:} \tag{5.24}$$

thus the kernel of $(A+B)$ is $(\mathcal{A}+\mathcal{B})$.

The kernel corresponding to matrix multiplication is derived as

$$(A \cdot B)\underline{x} = A \cdot [\mathcal{B} * X]_{:} = \Big[\mathcal{A} * \big[\,[\mathcal{B} * X]_{:}\big]_{n\times n}\Big]_{:} = [\mathcal{A} * \mathcal{B} * X]_{:} \tag{5.25}$$

thus the kernel of $(A \cdot B)$ is $(\mathcal{A} * \mathcal{B})$.

Because a scalar is unchanged when transposed, we can derive the kernel for L^T as follows:

$$\begin{array}{ccc} \underline{x}^T L\, \underline{y} & = (\underline{x}^T L\, \underline{y})^T = & \underline{y}^T L^T \underline{x} \\ \underline{x}^T [\mathcal{L} * Y]_{:} & = & \underline{y}^T \big[\mathcal{L}^T * X\big]_{:} \\ \sum_i x(i) \sum_j y(j) \mathcal{L}(i-j) & = & \sum_j y(j) \sum_i x(i) \mathcal{L}^T(j-i) \end{array} \tag{5.26}$$

from which it follows that a matrix transpose corresponds to a mirror image in the kernel domain:

$$\mathcal{L}(i) = \mathcal{L}^T(-i). \tag{5.27}$$

If a matrix is circulant (stationary, with periodic boundary conditions) then the kernel corresponding to the inverse matrix can be found using the FFT, as described in Section 8.3:

$$\mathcal{A}^{-1} = \text{FFT}_n^{-1}\big(1 \oslash \text{FFT}_n(\mathcal{A})\big). \tag{5.28}$$

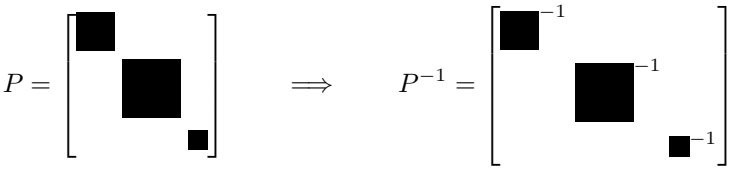
Since the FFT diagonalizes A, from the eigenvalues $\Lambda = \text{FFT}_n(\mathcal{A})$ the positive-definiteness of the associated matrix A can be tested as

$$A > \mathbf{0} \quad \text{iff} \quad \Lambda_{i,j} > 0 \;\; \forall i,j\,. \tag{5.29}$$

Matrix Inversion

We need to develop a skepticism any time large matrix inverses are called for. It is rare that an explicit inverse is needed and in most cases the problem can be reformulated.

However given a prior P, if the inverse P^{-1} is needed then direct inversion may be unavoidable. Most attractive is state decoupling (Chapter 8), leading to a block-diagonal matrix, in which case the matrix inverse P^{-1} is found by inverting each of the diagonal blocks individually:



Other solutions include state reduction (Chapter 8), series approximation (5.21), approximating the field as stationary and using kernel methods, or based on some sort of restricting assumption, such as a banded inverse [10] or a block-banded inverse [11]. The latter methods are assumption-dependent and have a sufficiently complex implementation to put them outside the scope of this text.

If the matrix is part of solving estimates, such as in

$$\underline{\hat{z}}(\underline{m}) = \left(L^T L + C^T R^{-1} C\right)^{-1} C^T R^{-1} \underline{m}, \tag{5.30}$$

then the inversion of $\left(L^T L + C^T R^{-1} C\right)^{-1}$ is avoidable. Instead, we should rewrite the problem as a linear system

$$\left(L^T L + C^T R^{-1} C\right) \underline{\hat{z}}(\underline{m}) = C^T R^{-1} \underline{m} \tag{5.31}$$

and solve this using Gaussian elimination, a Cholesky decomposition, or iterative methods, all of which are discussed in Chapter 9. The advantages of these latter approaches are in terms of computational complexity, storage complexity, and numerical robustness.

Complexity Comparison

Table 5.1 summarizes the computational complexity for the dense, banded, and kernel matrix types just discussed. The challenge in solving large multidimensional problems is not to find faster ways to invert huge covariance matrices, rather how to design or discover effective *models* which lead to attractive properties, such as sparsity, bandedness, block-diagonal matrices, diagonal dominance, or kernels.

Operation	Matrix Type		
	Dense	Banded	Kernel
Matrix Storage	$\mathcal{O}(n^2)$	$\mathcal{O}(bn)$	$\mathcal{O}(q)$
Matrix–Matrix Sum	$\mathcal{O}(n^2)$	$\mathcal{O}(bn)$	$\mathcal{O}(q)$
Matrix–Matrix Multiply	$\mathcal{O}(n^3)$	$\mathcal{O}(b^2n)$	$\mathcal{O}(q)$
Matrix–Vector Multiply	$\mathcal{O}(n^2)$	$\mathcal{O}(bn)$	$\mathcal{O}(qn)$
Matrix Inversion	$\mathcal{O}(n^3)$	$\mathcal{O}(n^3)$	$\mathcal{O}(nlogn)$
Positive-Definiteness Test	$\mathcal{O}(n^4)$	$\mathcal{O}(n^4)$	$\mathcal{O}(nlogn)$

Table 5.1. A comparison of computational complexity for dense, banded, and kernel matrix operations. In all cases the underlying problem $\underline{z}$ has n elements, banded matrices have b bands, and kernels have q nonzero elements.

5.4 Modelling

The previous section discussed mostly pragmatic matters — how to store and mathematically manipulate large matrices. However, this discussion is largely abstract and artificial, playing games with matrices, unless it can be connected, in a meaningful fashion, to the solution of large inverse problems of interest.

The challenge of modelling is not just one of specifying relationships, such as the N^6 interrelationships among the elements of an $N \times N \times N$ cube; it is fairly straightforward, after all, to invent or guess functions which assert a statistical relationship, such as an autocorrelation for a stationary model —

$$f(i,j,k) = E\big[z(0,0,0)z(i,j,k)\big]. \tag{5.32}$$

But not all choices of f make sense, or are even valid. Rather, problem modelling simultaneously needs to satisfy four criteria:

1. The model must be legal or valid, meaning that derived or implied covariances P, R or covariance inverses P^{-1} must be positive-definite.
2. The model should not be overfitting training data.
3. Ideally, the structure of the model should be compatible with an efficient mathematical solution, aspects of which were discussed in Section 5.3.
4. The model must be faithful to the problem. To the extent that the model is an approximation, the errors which appear should be negligible or tolerable.

The above criteria are individually straightforward, but challenging as a whole: the most obvious models tend to be invalid, the efficient and elegant models tend to be inaccurate, and the accurate models tend to be computationally infeasible.

By far the most serious problem in model specification and development is that of positive-definiteness:

- If a prior covariance P is negative-definite, even to only a tiny degree, the estimation results can become completely meaningless, and may not necessarily bear any resemblance to an approximate solution.
- Given a hypothesized prior covariance P, verifying that it is positive-definite is extremely difficult, requiring the application of Sylvester's test (Appendix A.3) or computing all of the eigenvalues of P.

In other words, it is crucial that P be positive definite, and it is essentially impossible to check the positive-definiteness of P!

There do exist methods [48], known as covariance extension, which can form a positive-definite covariance given a few correlations. Unfortunately this problem remains unsolved in higher than one dimension, and is computationally inconceivable, in any event, for problems of significant size.

Additional challenges to the modelling process are the extra complications added by the lexicographic reordering of the pixel elements. Furthermore the choice of a suitable model can depend subtly on the problem context, thus the actual process by which a model is selected is highly creative and therefore difficult to describe in a structured, step-by-step fashion.

We focus on the construction of models which are known to be positive-definite, so that eigendecompositions are *not* required for validation, and then to choose among these available models for suitable choices in a given problem context.

The next two sections examine "deterministic" and "statistical" models, respectively. The former, following on Section 3.1, seeks to specify a set of constraints L or L^TL, as required by (3.32). The latter, following on Section 3.2, seeks to specify prior models, possibly P in the case of (3.65) or P^{-1} for (3.67).

From Section 3.2.4 we know, of course, that there are both conceptual and algebraic overlaps between these two classes of models — that is, to some degree we can associate L^TL with P^{-1}. In particular, as will become clear later, there are relationships between the local deterministic models of Section 5.5 and the Markov statistical models of Section 5.6.4.

5.5 Deterministic Models

By deterministic modelling, we mean the assertion of a set of constraints which are compatible with the types of behaviour that are expected in $\underline{z}$, but we do not neces-

sarily believe that these constraints represent a complete description of the statistics of $\underline{z}$. That is, the constraints serve to regularize and condition the problem, not so much to model it.

We assert a set of constraints $\{\underline{l}_i^T \underline{z}\}$, equivalently penalizing

$$\|L\underline{z}\| = \underline{z}^T L^T L\, \underline{z}, \qquad \text{where } L = \begin{bmatrix} \underline{l}_1^T \\ \vdots \\ \underline{l}_q^T \end{bmatrix}. \tag{5.33}$$

Thus L is a rectangular matrix, where the number of columns equals n, the dimensionality of $\underline{z}$, and where the number of rows q equals the number of constraints. Indeed, the ordering of constraints (rows) in L is completely arbitrary, which is both intuitive and easily verified by seeing that $L^T L$ is unaffected by row-reordering in L.

The key to this approach is its guaranteed positive-semidefiniteness. Recalling the definition of a positive-semidefinite matrix Q:

$$Q \geq \mathbf{0} \qquad \Longleftrightarrow \qquad \underline{x}^T Q\, \underline{x} \geq 0 \;\; \forall \underline{x}, \tag{5.34}$$

thus

$$\underline{x}^T L^T L\, \underline{x} = (L\underline{x})^T (L\underline{x}) = \underline{y}^T \underline{y} \geq 0 \;\; \forall \underline{y}. \tag{5.35}$$

That is, *any* real matrix product $L^T L$ is, by definition, positive-semidefinite.

The most common choices for L involve *local* constraints; that is, each constraint is a function of only a few spatially proximate elements of $\underline{z}$. Although locality is intuitively appealing it is not of great significance computationally; indeed, any locality in two- and higher-dimensional problems is quickly destroyed by the lexicographic reordering. Rather, the key is that each constraint be sparse, involving only a small number of elements of $\underline{z}$, whether spatially local nor not.

The starting point for L is normally the discretization of simple variational constraints, which were introduced in Section 2.4.1:

$$\text{Membrane:} \qquad \|L\underline{z}\| = \iint \left|\frac{\partial z}{\partial x}\right|^2 + \left|\frac{\partial z}{\partial y}\right|^2 dx\, dy \tag{5.36}$$

$$\simeq \|L_x \underline{z}\| + \|L_y \underline{z}\|, \tag{5.37}$$

$$\text{Thin Plate:} \qquad \|L\underline{z}\| = \iint \left|\frac{\partial^2 z}{\partial x^2}\right|^2 + 2\left|\frac{\partial^2 z}{\partial x \partial y}\right|^2 + \left|\frac{\partial^2 z}{\partial y^2}\right|^2 dx\, dy \tag{5.38}$$

$$\simeq \|L_{xx} \underline{z}\| + 2\|L_{xy} \underline{z}\| + \|L_{yy} \underline{z}\|, \tag{5.39}$$

where $L_x, L_y, L_{xx}, L_{xy}, L_{yy}$ represent first- and second-order difference constraints approximating the desired partial derivatives. For example, returning to the 3×3 example of (5.1),

z_{11}	z_{12}	z_{13}
z_{21}	z_{22}	z_{23}
z_{31}	z_{32}	z_{33}

there are a total of six first-order horizontal and six first-order vertical constraints:

$$\underbrace{\begin{bmatrix} \text{-1} & & 1 & & & & \\ & \text{-1} & & 1 & & & \\ & & \text{-1} & & 1 & & \\ & & & \text{-1} & & 1 & \\ & & & & \text{-1} & & 1 \\ & & & & & \text{-1} & & 1 \end{bmatrix}}_{L_x} \quad \underbrace{\begin{bmatrix} \text{-1} & 1 & & & & & \\ & \text{-1} & 1 & & & & \\ & & & \text{-1} & 1 & & \\ & & & & \text{-1} & 1 & \\ & & & & & \text{-1} & 1 \end{bmatrix}}_{L_y} \tag{5.40}$$

The band spacing in L_x and the staggered bands in L_y are due to the lexicographic structure of the one-dimensional vector $\underline{z}$. The constraints are much more easily interpreted if the rows of L are lexicographically unwound:

$$(\underline{l}_x)_1^T = \begin{bmatrix} \text{-1} & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \Longrightarrow \left[(\underline{l}_x)_1^T\right]_{3\times 3} = \begin{bmatrix} \text{-1} & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tag{5.41}$$

a difference constraint on two horizontally adjacent elements, and

$$(\underline{l}_y)_2^T = \begin{bmatrix} 0 & \text{-1} & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \Longrightarrow \left[(\underline{l}_y)_2^T\right]_{3\times 3} = \begin{bmatrix} 0 & 0 & 0 \\ \text{-1} & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \tag{5.42}$$

a difference constraint on two vertically separated ones. Note that if a given constraint set, such as L_x, is stationary, then it is much simpler and more intuitive to think in terms of its generator or kernel:

$$L_x \underline{z} = (\mathcal{L}_x * Z)_{:} \tag{5.43}$$

a two-dimensional convolution, in which case $L_x, L_y, L_{xx}, L_{yy}, L_{xy}$ are represented more compactly and elegantly via their corresponding kernels:

$$\mathcal{L}_x = \begin{bmatrix} \text{-1} & 1 \end{bmatrix} \qquad \mathcal{L}_y = \begin{bmatrix} \text{-1} \\ 1 \end{bmatrix} \tag{5.44}$$

$$\mathcal{L}_{xx} = \begin{bmatrix} \text{-1} & 2 & \text{-1} \end{bmatrix} \qquad \mathcal{L}_{yy} = \begin{bmatrix} \text{-1} \\ 2 \\ \text{-1} \end{bmatrix} \qquad \mathcal{L}_{xy} = \begin{bmatrix} \text{-1} & 1 \\ 1 & \text{-1} \end{bmatrix} \tag{5.45}$$

Rather than expressing the terms $L_x, L_y, \ldots$ separately, we can consider the complete set of constraints

$$L = \begin{bmatrix} L_x \\ L_y \end{bmatrix} \qquad \text{or} \qquad L = \begin{bmatrix} L_{xx} \\ \sqrt{2} L_{xy} \\ L_{yy} \end{bmatrix} \tag{5.46}$$

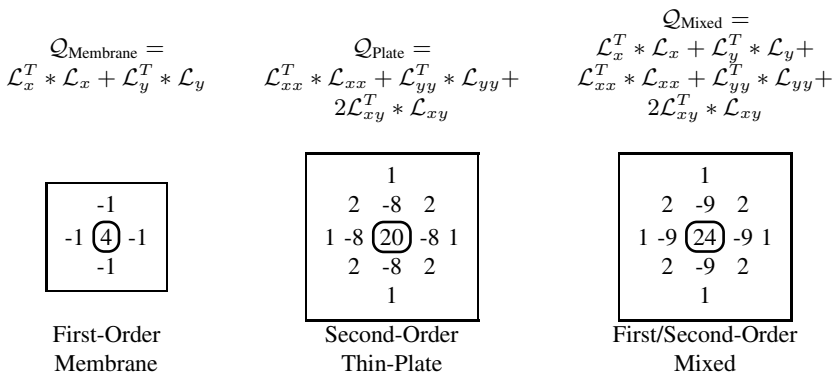


Fig. 5.8. Three typical constraint kernels $\mathcal{Q}$, derived from the given first- and second-order pixel constraints. The manner of the derivation is shown in (5.49).

for the first and second-order cases, respectively, although obviously other choices of L are possible. In solving the estimation problem (3.32) we need to represent $Q = L^T L$. In some cases, particularly for stationary problems of modest size, it may be more convenient to store Q and not L itself. Given a number of constraints $L_1, L_2, \ldots$, we may find Q as

$$L = \begin{bmatrix} L_1 \\ L_2 \\ \vdots \end{bmatrix} \quad \Longrightarrow \quad Q = L^T L = \sum_i L_i^T L_i. \tag{5.47}$$

Analogously to before, if the constraints are stationary then we can consider the more intuitive convolutional kernel representation $\mathcal{Q}$ for Q, such that

$$Q = \sum_i L_i^T L_i \quad \Longrightarrow \quad Q\underline{z} = (\mathcal{Q} * Z)_{:} = \left(\left(\sum_i \mathcal{L}_i^T * \mathcal{L}_i \right) * Z \right)_{:} \tag{5.48}$$

As a simple, two-dimensional example,

$$\begin{aligned} \mathcal{Q}_{\text{Membrane}} = \mathcal{L}_x^T * \mathcal{L}_x + \mathcal{L}_y^T * \mathcal{L}_y &= \begin{bmatrix} 1 & -1 \end{bmatrix} * \begin{bmatrix} -1 & 1 \end{bmatrix} + \begin{bmatrix} 1 \\ -1 \end{bmatrix} * \begin{bmatrix} -1 \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} -1 & 2 & -1 \end{bmatrix} + \begin{bmatrix} -1 \\ 2 \\ -1 \end{bmatrix} = \begin{bmatrix} & -1 & \\ -1 & 4 & -1 \\ & -1 & \end{bmatrix}. \end{aligned} \tag{5.49}$$

Two further kernels, computed in the same way, are shown in Figure 5.8.

As discussed in Chapter 9 (see also Problem 5.5), it is important to realize that for very large, nonstationary problems it may be preferable to store L and to leave Q implicit, such that the product $Q\underline{z}$ is evaluated as

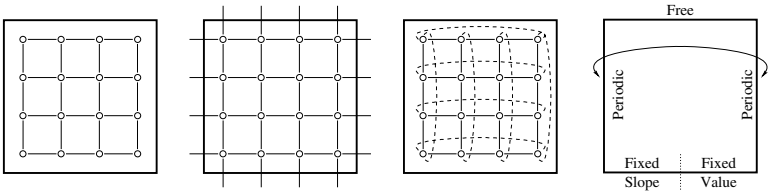


Fig. 5.9. Boundary conditions are implicitly specified by the presence or absence of constraints for pixels at or near the domain boundary. Four typical examples are shown here of a 4×4 pixel domain with constraints shown as lines. Specific examples of kernels are shown in Figures 5.10 and 5.11.

$$Q\underline{z} = L^T L\underline{z} = \left(L^T\left(L\underline{z}\right)\right). \tag{5.50}$$

The reason for preferring L is that it is typically sparser than Q, and so more efficient to represent. Furthermore, once L has been specified, there is no compelling reason to explicitly calculate the matrix–matrix product $Q = L^T L$. Finally, if in trying to avoid the matrix–matrix product we try to specify Q directly, there are challenges in guaranteeing its positive-definiteness.

5.5.1 Boundary Effects

For any field of finite extent, it is required to specify the behaviour of the field at its boundaries. The details of these boundary conditions will be highly problem-dependent, but will normally fall into one of the categories shown in Figure 5.9:

FREE BOUNDARY: All constraints which would refer to pixels outside of the lattice are removed from L, therefore the boundary asserts no constraint (the state element is "free"). With these constraints removed the resulting kernel becomes nonstationary and varies in a predictable fashion, as illustrated in Figure 5.10.

ZERO BOUNDARY: In contrast to the free boundary, above, if all constraints which would refer to pixels outside of the lattice are *truncated*, as illustrated in Figure 5.11, then the truncated portions of the constraints *implicitly* set the state values outside of the boundary to zero.

SPECIFIED BOUNDARY: In some cases we may wish to assert boundary values other than zero, for example to specify the gradient at the boundary, as illustrated in Figure 5.12. Let B be a specified set of boundary values around random field Z.

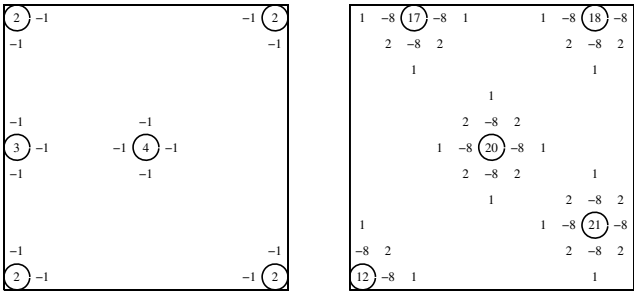


Fig. 5.10. The free boundary kernel: the squared constraints ($\mathcal{Q}$) relating to pixels outside of the domain have been removed, resulting in a nonstationary kernel. In all cases the circled element denotes the kernel origin. Compare with the boundaries in Figure 5.11.

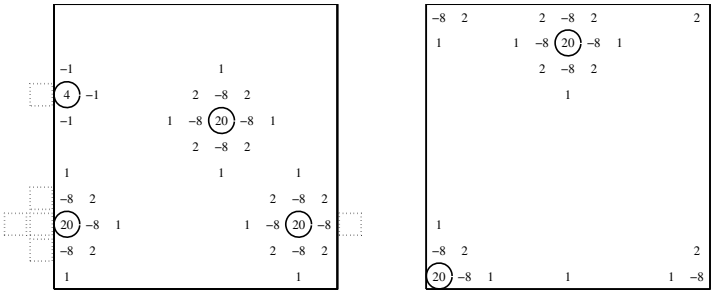


Fig. 5.11. Kernels at boundaries: if the kernel $\mathcal{Q}$ is truncated (left), then there is an implicit reference to zero-valued pixels, imposing a boundary condition. If the kernel $\mathcal{Q}$ is wrapped (right), then the surface is constrained to be periodic.

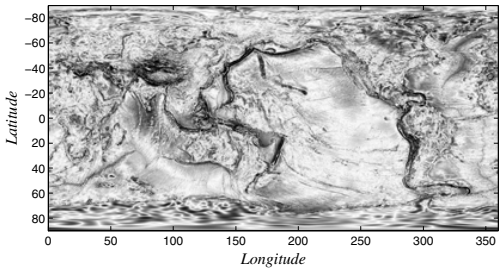
Then a given constraint $\underline{l}$, which spills from the random field into the boundary, is asserted as

$$\left\| \underline{l}^T \begin{bmatrix} \underline{z} \\ \underline{b} \end{bmatrix} \right\|. \tag{5.51}$$

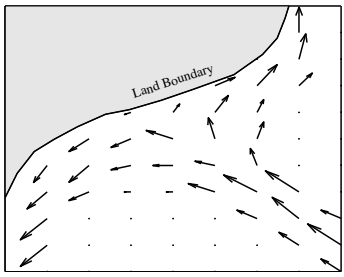
By separating the portions of the constraint applied to Z and B we can rewrite the penalty (5.51) as

$$\left\| \begin{bmatrix} \underline{l}_z^T & \underline{l}_b^T \end{bmatrix} \begin{bmatrix} \underline{z} \\ \underline{b} \end{bmatrix} \right\| = \left\| \underline{l}_z^T \underline{z} + \underline{l}_b^T \underline{b} \right\| = \left\| \underline{l}_z^T \underline{z} + \beta \right\|. \tag{5.52}$$

Collecting all of the constraints into matrix form, we have the net penalty



Global Gravitational Equipotential:
Periodic East/West Boundaries
(Data Source NGA Office of GEOINT Sciences)



Fluid Flow at a Coastline:
Zero Flow Normal to Boundary

Fig. 5.12. Boundaries appear very frequently in remote sensing. The *Geoid*, left, has east and west sides adjacent (periodic), whereas the top (north) and bottom (south) edges are not periodic. Fluid flow near a coastline, right, is constrained to have a zero flow component at right angles to the coast.

$$\left\| L_z^T \underline{z} + \underline{\beta} \right\|, \tag{5.53}$$

so we see that the imposition of specified boundary conditions appears similarly to that of a prior mean; the solution of (5.53) is discussed in parallel with the discussion of prior means in Section 5.5.3.

PERIODIC OR WRAPPED BOUNDARY: State elements at opposing edges (top / bottom, left / right, etc.) are treated as adjacent, illustrated in Figure 5.11. Thus a reference to a pixel outside of the lattice is wrapped back inside, so for a $K \times N$ lattice,

$$z(i, j) \equiv z(i \bmod K, j \bmod N). \tag{5.54}$$

Although such models are rarely perfectly realistic, they are popular because of the simplicity and computational efficiency of stationary models. Having left–right periodic boundaries is very common when estimating on a sphere, such as global remote sensing of the Earth, as shown in Figure 5.12.

Clearly the above boundary types can be combined, in that part of the boundary may be free, another part may be zero, two sides may be periodic, etc.

5.5.2 Discontinuity Features

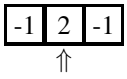
With the exception of the boundaries we have, thus far, implicitly assumed stationary constraints. Although the *inference* of a nonstationary model can be quite difficult,

and beyond the scope of this discussion, the *assertion* of known nonstationarities in a model is straightforward.

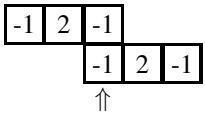
Indeed, the constraints at each state element should reflect our understanding of smoothness (or not) at each location. As long as they can be encoded in terms of linear constraints, there are no restrictions on the types of nonstationarities to be asserted. A few examples follow, in which a single one-dimensional constraint L is illustrated, although the concepts are equally applicable to 2D and 3D. In each case the stated property occurs at the point marked by an arrow:

First-order smoothness (no abrupt change in value) between the marked elements.

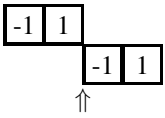
Second-order smoothness (no abrupt change in slope) at the marked state element.



State elements to the left and right are smooth, however a fold (abrupt change in slope) may occur at the marked state element. That is, the value of the marked element is constrained to be consistent with those of its neighbours, however there are *no* constraints on its slope.



The two state elements to the left and to the right are constrained to be similar, however between the marked elements a jump discontinuity (an abrupt change in value) may occur, since there are no constraints relating the corresponding state values.



The use of the preceding elements is illustrated in Figure 5.13.

5.5.3 Prior-Mean Constraints

The standard membrane (first-order) and thin-plate (second-order) models consist strictly of terms constraining the *relative* values of proximal pixels, but no constraint of any kind on the value of any single pixel. Thus from the perspective of the regularizing constraints L, the overall image mean is irrelevant:

$$\|L\underline{z}\| \equiv \|L(\mu\underline{1} + \underline{z})\| \tag{5.55}$$

for any constant, scalar mean μ. In other words, the overall image mean lies in the nullspace of L, thus clearly L^TL is singular.

To remove the singularity of L^TL, or in those cases where we wish to assert a prior mean, we need to introduce a constraint on an individual state element or, more commonly, to introduce a constraint on each element. Given a prior mean $\underline{\mu}$ for each

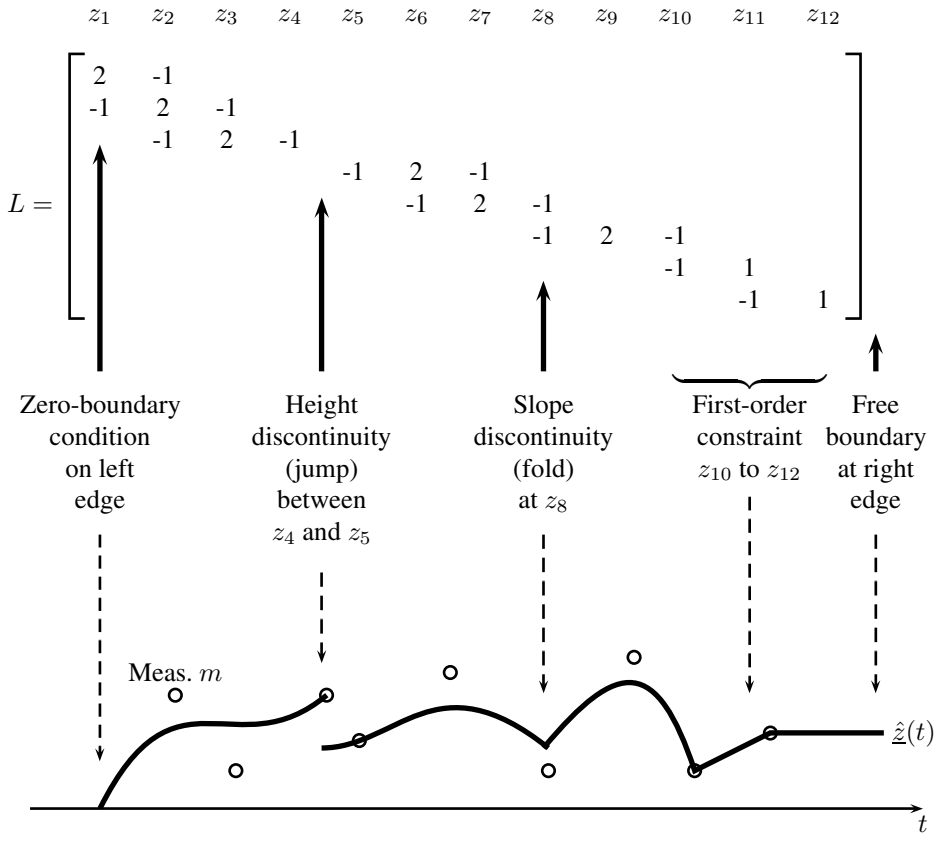


Fig. 5.13. A detailed example of boundary conditions and first- / second-order constraints. For simplicity, the constraints L are illustrated for a one-dimensional example, so there are no lexicographic effects. Clearly L describes the constraints for a twelve-element state $\underline{z}$, so the sketch in the lower panel is essentially a cartoon, a continuous one-dimensional estimated signal, consistent with the constraints and the nine measurements (open circles).

element in $\underline{z}$ and possible specified boundary conditions $\underline{\beta}$ from (5.53), we wish to penalize

$$\left\| \begin{bmatrix} L \\ \alpha I \end{bmatrix} \underline{z} - \begin{bmatrix} \underline{\beta} \\ \underline{\mu} \end{bmatrix} \right\|, \tag{5.56}$$

leading to a weighted least-squares formulation

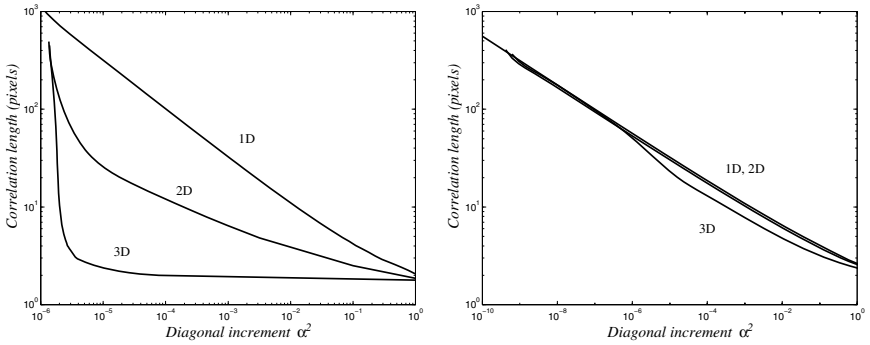
$$\hat{\underline{z}} = \left(C^T W C + L^T L + \alpha^2 I \right)^{-1} \left(C^T W \underline{m} + L^T \underline{\beta} + \underline{\mu} \right). \tag{5.57}$$

The addition of diagonal constraints corresponds to increasing the value of the central element in the kernel domain, as shown in Figure 5.14. As α increases the variance

$$\begin{array}{ccc} & -1 & \\ -1 & \boxed{4+\alpha^2} & -1 \\ & -1 & \end{array} \qquad \begin{array}{ccccc} & & 1 & & \\ & 2 & -8 & 2 & \\ 1 & -8 & \boxed{20+\alpha^2} & -8 & 1 \\ & 2 & -8 & 2 & \\ & & 1 & & \end{array}$$

First-Order Membrane | Second-Order Thin-Plate

Fig. 5.14. To constrain the value of a pixel, either to limit its variance or to assert a prior mean, we can choose to introduce a zeroth-order constraint, increasing the central element of the kernel by some amount α^2.



(a) Membrane Prior (b) Thin-Plate Prior

Fig. 5.15. The effect of asserting a prior mean: for the two common smoothness models under consideration, as α^2 in Figure 5.14 increases the process correlation length is decreasing.

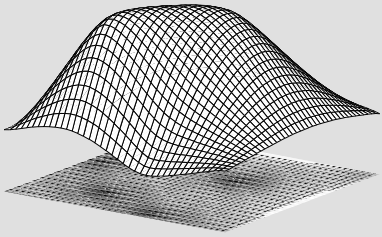
of each state element decreases, as does the spatial correlation length. Figure 5.15 plots this correlation length for the membrane and thin-plate priors in one, two, and three dimensions.

5.6 Statistical Models

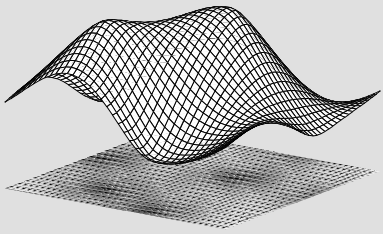
The previous discussion focused on deterministic models for regularization. Because these models are mostly heuristically chosen, and guaranteed to be positive-semidefinite, the discussion was straightforward and mostly intuitive.

Example 5.1: Multidimensional Interpolation

We consider a two-dimensional estimation problem. We are given four measurements, and choose to assert second-order (thin-plate) smoothness by asserting the kernels $\mathcal{L}_{xx}, \mathcal{L}_{xy}, \mathcal{L}_{yy}$ of (5.45) at each pixel. Only the boundaries remain to be specified; the following figures show the estimation results for the cases of free and periodic boundaries:



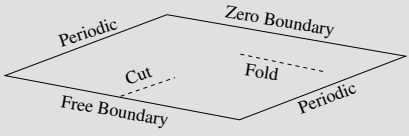
Free Boundaries



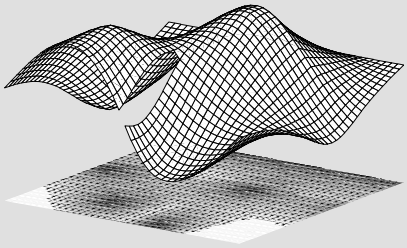
Periodic Boundaries

The surfaces show the resulting estimates, with the underlying images plotting the estimation error variances, clearly identifying the four measurement locations by their low variances (dark). Note the response in the error variances on the right-hand side of the periodic surface to the measurements on the opposite (left) side of the domain, compared to the absence of such a response for the free boundary.

A much richer example is constructed below, consisting of two periodic boundaries, one free boundary, one zero boundary, a cut (zeroth-order discontinuity) and a fold (first-order discontinuity):



Domain Structure



Estimation Results

One can observe how the reduction or elimination of constraints by the the cut and fold leads to a corresponding increase (white) in the error variances. Because the zero-boundary condition is a strong constraint, it greatly limits the variability of the surface and thus leads to a low error variance (dark, back-right), in contrast to the opposing free boundary with high error variance (light, front-left).

For the remainder of the chapter our focus shifts considerably to specifying statistical prior models. We realize that the constraints L *implicitly* specify the statistics of $\underline{z}$, by identifying $L^T L$ with P^{-1}. However, in the implicitness of the specification lies the problem: given a random field Z, with known or sample statistics, it is not clear how to practically formulate the associated L. Algebraically the formulation is simple:

$$P \quad \Leftrightarrow \quad P^{-1} = V D V^T \quad \Leftrightarrow \quad L = D^{1/2} V^T. \tag{5.58}$$

That is, we solve for the matrix square root L, computationally impossible for all but the most trivially-sized problems. Furthermore, an arbitrarily chosen P and the derived L may not permit compact or efficient representation and storage.

Even the direct specification of P carries with it the burden of guaranteeing positive-definiteness, so there remains an interest in implicit statistical methods which specify the model through the covariance-inverse square root L, particularly so for nonstationary problems (see Problem 5.5, for example).

The following four sections summarize methods for prior model specification, with ideas developed in greater detail in later chapters as appropriate. The first two models are *explicit*, specifying P; the latter two are *implicit*, specifying L or P^{-1}.

- Section 5.6.1: Dense P — Analytical positive-definite forms
- Section 5.6.2: Dense P — Positive-definite forms for nonstationary fields
- Section 5.6.3: Sparse Γ, A — Square root / dynamic models
- Section 5.6.4: Sparse P^{-1} — Banded inverse-covariance models

5.6.1 Analytical Forms

Consider setting up an estimation problem in d dimensions, characterized by a stationary correlation function

$$E\big[z(\underline{x})z(\underline{x}+\underline{\delta})\big] - E\big[z(\underline{x})\big]E\big[z(\underline{x}+\underline{\delta})\big] = \sigma^2 f\,(r)\,, \qquad r^2 = \sum_{i=1}^{d} \delta_i^2. \tag{5.59}$$

That is, $\underline{x}$ and $\underline{\delta}$ are a point and offset, respectively, in d-dimensional space, and r measures the Euclidean length of offset $\underline{\delta}$. So in two dimensions, the statistics are characterized by a stationary correlation matrix

$$E\big[z(0,0)z(x,y)\big] - E\big[z(0,0\big]E\big[z(x,y)\big] = \mathcal{P}(x,y). \tag{5.60}$$

There are only a few analytic forms for f which are guaranteed to be positive-definite [76, 162, 163, 248], plotted in Figure 5.16. Each is controlled by a single parameter θ, which determines the range or correlation-length of the function.

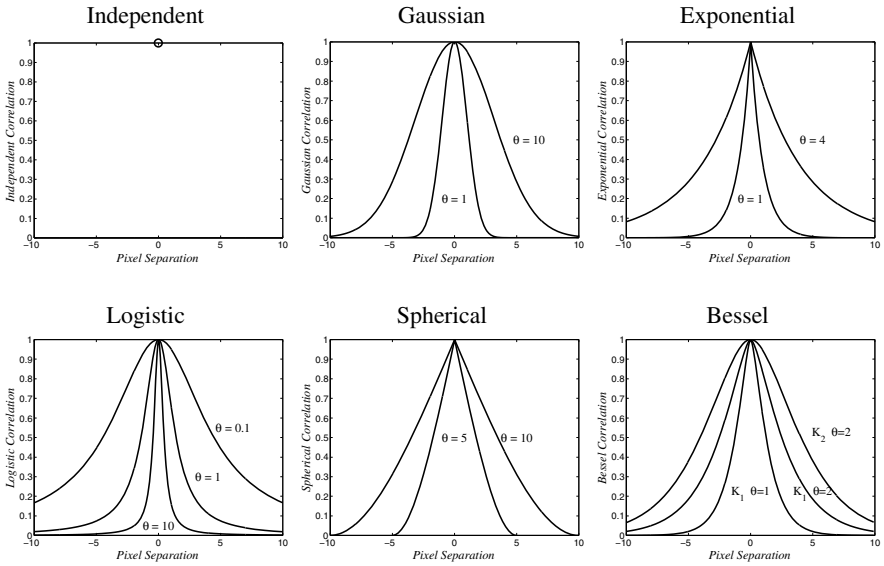


Fig. 5.16. Six common positive-definite forms, shown for two or three values of width parameter θ. The smoothness, numerical conditioning, and spatial extent of these are very different from one to the next, as summarized in Table 5.2.

EXPONENTIAL:

$$f_{\text{Exp}}(r,\theta) \triangleq \exp\left(-\frac{|r|}{\theta}\right) \tag{5.61}$$

Very well conditioned numerically, even for highly-correlated (large θ) domains, however the sharpness of the peak at the origin causes the locations of sparse measurements to induce corresponding peaks in the resulting estimates, normally an undesirable artifact.

SPHERICAL:

$$f_{\text{Sph}}(r,\theta) \triangleq \begin{cases} 1 - \frac{3}{2}\left(\frac{r}{\theta}\right) + \frac{1}{2}\left(\frac{r}{\theta}\right)^3 & r \le \theta \\ 0 & r > \theta \end{cases} \tag{5.62}$$

Similar conditioning and peak-sharpness properties to the exponential case, however the spatial extent of the correlation is strictly limited to a finite range $r \le \theta$.

GAUSSIAN:

$$f_{\text{Gau}}(r,\theta) \triangleq \exp\left(-\frac{r^2}{2\theta}\right) \tag{5.63}$$

Extremely smooth, but also leading to highly ill-conditioned covariance matrices.

LOGISTIC:

$$f_{\text{Log}}(r,\theta) \triangleq 1 - \frac{\theta r^2}{2+\theta r^2} \tag{5.64}$$

Similar to the Gaussian case — smooth and ill-conditioned — but faster to compute.

BESSEL:

$$f_{\text{Bes}}(r,\theta) \triangleq \frac{(2r/2\theta)^\nu}{\Gamma(\nu)} K_\nu(r/\theta) r K_1(r/\theta) \tag{5.65}$$

A whole family of correlation functions, corresponding to the statistics of various orders of Laplacians, the Bessel correlations are smooth and very well conditioned, but of much greater computational complexity. The first-order case is comparatively simple and worth noting:

$$f_{\text{Bes}}(r,\theta) \triangleq \frac{r}{\theta} K_1\left(\frac{r}{\theta}\right). \tag{5.66}$$

Separate from the preceding five functions there are two special cases, corresponding to the limiting behaviour of the above functions as $\theta \to 0$ and $\theta \to \infty$:

INDEPENDENT:

$$f_{\text{Ind}}(r) \triangleq \begin{cases} 1 & r = 0 \\ 0 & r > 0 \end{cases} \tag{5.67}$$

As $\theta \to 0$ each element becomes uncorrelated with every other. Normally not a useful model on its own, however see the discussion below regarding the nugget effect.

CONSTANT:

$$f_{\text{Const}}(r) \triangleq 1 \tag{5.68}$$

As $\theta \to \infty$, all of the state elements become perfectly correlated; that is, the entire random field is constant. The model is positive-semidefinite and not useful on its own, however see the kriging discussion, below.

A summary of the properties of the positive-definite forms is shown in Table 5.2, with a short example in Figure 5.17.

Finally, because the positive linear combination of positive-definite matrices is itself positive-definite:

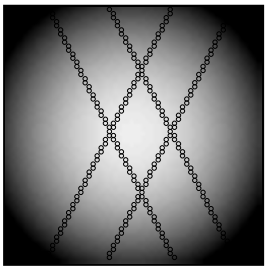
$$A, B > \mathbf{0} \qquad \Longrightarrow \qquad \alpha A + \beta B > \mathbf{0} \;\text{ for } \alpha, \beta > 0 \tag{5.69}$$

therefore any linear combination of the above correlation functions remains positive-definite. Of particular interest is the common form

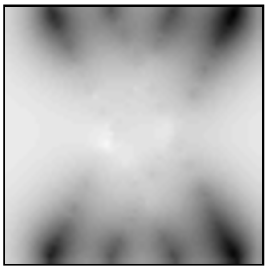
$$f_{\text{Nugget}}(r) = \alpha f_{\text{Ind}}(r) + \beta f_*(r) \tag{5.70}$$

Method	Smoothness	Extent	Conditioning	Complexity
Independent	Bad	Zero	Excellent	Trivial
Constant	Good	Infinite	Singular	Trivial
Exponential	Poor	Infinite	Good	OK
Spherical	Poor	Finite	Good	Easy
Bessel	Good	Infinite	Good	Hard
Logistic	Good	Infinite	Poor	Easy
Gaussian	Good	Infinite	Bad	OK

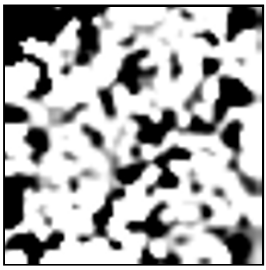
Table 5.2. A summary comparison of qualitative properties of the analytical positive-definite forms. None of these can be said to be objectively superior to the others.



Measurement Locations on True Field



Exponential f_{Exp}



Gaussian f_{Gau}

Spherical f_{Sph}

Fig. 5.17. Three illustrations of spatial estimation based on analytical forms, based on the scanning pattern of satellite measurements (as in Figure 1.3 on page 5). The Gaussian results are numerically unstable due to ill-conditioning; the spherical results suffer from a local correlation; the exponential results are reasonably credible, however in general the exponential is like a first-order (membrane) prior and provides insufficient smoothing.

for any valid correlation f_*, which models the so-called "nugget" effect of the kriging literature [75,76] (see Section 2.5.3), in which there is a certain degree of instantaneous decorrelation (controlled by α) between a random element and any other, regardless of how close. That is, the random field contains a white component.

Of interest is also the other extreme, in which a constant is added to the correlation

$$f_{\text{Weak Prior}}(r) = \alpha f_{\text{Const}}(r) + \beta f_*(r) \tag{5.71}$$

for any valid correlation f_*. It is clear that the prior variance

$$\text{var}\big(z(\underline{x})\big) = \sigma^2 f_{\text{Weak Prior}}(0) = \sigma^2(\alpha + \beta) \tag{5.72}$$

increases as α increases, meaning that the prior mean is asserted ever more weakly. On the other hand, the statistics of pixel *differences*

$$\begin{aligned} \text{var}\big(z(\underline{a}) - z(\underline{b})\big) &= \sigma^2\big[2f_{\text{Weak Prior}}(0) - 2f_{\text{Weak Prior}}(|\underline{a} - \underline{b}|)\big] \\ &= 2\beta\sigma^2\big[f_*(0) - f_*(|\underline{a} - \underline{b}|)\big] \end{aligned} \tag{5.73}$$

are independent of α. Thus our spatial statistics are left untouched; only the prior mean is increasingly ignored. As $\alpha \to \infty$, the solution to the estimation problem becomes equivalent to kriging, however the poor numerical conditioning associated with large α makes such an approach impractical.

The key problem in directly modelling the covariance of random fields is that the above correlation functions do *not* generalize to the nonstationary case. That is, using any of the above correlations with a space-varying θ normally leads to covariances which fail to be positive-semidefinite.

5.6.2 Analytical Forms and Nonstationary Fields

Arbitrary spatial variations in θ are not possible, since the resulting covariances fail to be positive-definite; however there are certain spatial variations which *are* permitted.

Independent Blocks

If the spatial nonstationarities occur in disjoint, independent regions, each of which is stationary, then the overall covariance is positive-definite. Specifically, if field Z on lattice Ω is divided into regions $\{R_1, \ldots, R_q\}$ such that

$$R_i \cap R_j = \emptyset \;\; i \neq j \qquad \cup_i R_i = \Omega \tag{5.74}$$

then if we lexicographically order by region, the resulting field covariance is block-diagonal:

$$\underline{z} = \begin{bmatrix} [Z(R_1)]_: \\ \vdots \\ [Z(R_q)]_: \end{bmatrix} \sim \begin{bmatrix} \blacksquare & & \\ & \blacksquare & \\ & & \blacksquare \end{bmatrix} \tag{5.75}$$

If each of the blocks is positive-definite, then so is the block-diagonal matrix. This approach is, however, of very limited interest, since most images and random fields are not piecewise-independent.

Spatially Varying Variance

If the nonstationarities are confined to spatial variations in the random field *variances*, and if we select a positive-definite, stationary correlation-coefficient structure $f()$, then the model remains positive-definite. That is, if we define a stationary field

$$\underline{z}^{(s)} \sim P > \mathbf{0} \quad \text{where} \quad P_{ij} = f(|i-j|) \tag{5.76}$$

and then define a nonstationary field $\underline{z}^{(ns)}$, where each element of $\underline{z}^{(ns)}$ is a rescaled version of $\underline{z}^{(s)}$:

$$\underline{z}^{(ns)} = \text{Diag}(\underline{\sigma})\underline{z}^{(s)} = D\underline{z}^{(s)} \sim DPD^T > \mathbf{0}. \tag{5.77}$$

The resulting covariance is thus made up of nonstationary (variance) and stationary (correlation) components:

$$E\big[z(\underline{x})z(\underline{x}+\underline{\delta})\big] - E\big[z(\underline{x})\big]E\big[z(\underline{x}+\underline{\delta})\big] = \underbrace{\sigma_{\underline{x}}\sigma_{\underline{x}+\underline{\delta}}}_{\text{Nonstationary}} \cdot \underbrace{f\left(|\underline{\delta}|\right)}_{\text{Stationary}}. \tag{5.78}$$

Alternately, we can explicitly write the covariance of the field as

$$P = \left(\underline{\sigma}\,\underline{\sigma}^T\right) \odot P_o, \tag{5.79}$$

where we see P expressed as a stationary structure P_o, modulated by nonstationary standard deviations.

Nonstationary Mixtures of Stationary Fields

In many situations the correlation structure f is itself nonstationary, in which case modelling is somewhat more difficult. Such situations commonly occur in recursive dynamic estimation in which measurements are assimilated in multiple steps over time.

Given a correlation structure $f(r,\theta)$, if θ varies with spatial location, the resulting covariance is almost certainly not positive-definite. However, it is possible to use a nonstationary mixture of stationary models [191], an approach most effective if

the modelled correlation structure has a single form (e.g., exponential), but with a space-varying correlation length θ. That is, for each unknown element z_j, suppose we have an associated idealized parameter setting θ_j. The idea is to generate q sets of estimates $\left\{\underline{\hat{z}}(\underline{m}|\Theta_i)\right\}$ and possibly error covariances $P(\underline{m}|\Theta_i)$, each based on a *stationary* positive-definite model f characterized by one of $\Theta_1, \ldots, \Theta_q$.

The desired *non*stationary estimates and error variances are then found as a nonstationary linear combination of the stationary ones; that is,

$$\hat{z}_j = \sum_i \alpha_i(\theta_j)\hat{z}_j(\underline{m}|\Theta_i) \tag{5.80}$$

$$P_{jj} = \sum_i \beta_i(\theta_j)P_{jj}(\underline{m}|\Theta_i). \tag{5.81}$$

The interpolating weights α_i, β_i satisfy

$$\sum_i \alpha_i(\theta) = 1 \qquad \sum_i \beta_i(\theta) = 1 \qquad \forall\theta \tag{5.82}$$

to prevent biasing the estimates, and

$$\alpha_i(\Theta_k) = \begin{cases} 1 & i = k \\ 0 & i \neq k \end{cases} \qquad \beta_i(\Theta_k) = \begin{cases} 1 & i = k \\ 0 & i \neq k \end{cases} \tag{5.83}$$

to require the exact solution to emerge when the desired θ equals one of the $\{\Theta_i\}$. The weights themselves can be solved by least squares to minimize the estimation error [191]. A brief illustration of this method can be seen in Application 4 on page 122.

5.6.3 Recursive / Dynamic Models

The previous two sections examined the explicit construction of a covariance P, which has the advantage of specifying an unambiguous statistical model, but the disadvantage of being a dense matrix for which positive-definiteness must be guaranteed.

However, in the same way that the constraint matrix L, the matrix square root of the system matrix Q, has no positivity requirements, the same is true of the matrix square root for the field covariance P or correlation structure $\mathcal{P}$. In particular, we could specify a random field $\underline{z}$ as

$$\underline{z} = [\mathbf{\Gamma} * W]_{:} \quad \Rightarrow \quad \mathcal{P} = \mathbf{\Gamma} * \mathbf{\Gamma}^T \tag{5.84}$$

$$\underline{z} = \Gamma\underline{w} \quad \Rightarrow \quad P = \Gamma\Gamma^T, \tag{5.85}$$

where $\underline{w}$ and W are white unit-variance random fields.

Such a representation indeed frees us from positive-definiteness requirements, however Γ remains large and dense and, in contrast to P, is difficult to interpret or to specify. Computing Γ directly from P may be impractical for large problems, depending on the sparsity of P (see Section 9.1.2), because of the computational complexity associated with finding matrix square roots.

A more efficient alternative is a recursive-dynamic model for the square root. Given a dynamic model

$$\underline{z}(t+1) = A\underline{z}(t) + B\underline{w}(t), \tag{5.86}$$

or possibly its space-stationary variant,

$$Z(t+1) = \mathcal{A} * Z(t) + \mathcal{B} * W(t), \tag{5.87}$$

plus an initial condition $\mathrm{cov}\big(\underline{z}(0)\big) = P_o$, then the process covariance is implicitly specified and guaranteed to be positive-definite.

Like L, A and B are typically sparse and subject to no other conditions.[4] Furthermore the Kalman smoother can be used to generate estimates $\hat{\underline{z}}(t|T)$ based on *all* of the measurements scattered throughout the dynamic process.

The challenge, clearly, is how to determine a partitioning of the problem such that the partitions obey a dynamic relationship. That is, given a multidimensional random field Z, how can we define partitions $\underline{z}(t) = \Xi_t Z_{:}$ such that the resulting dynamics on $\underline{z}(t)$ capture the statistics of Z?

The clear benefit of such a scheme, of course, is that the individual states $\underline{z}(t)$ are much smaller than the entire process $Z_{:}$, thus the size of dynamic A is much, much smaller than the single constraint matrix L which it replaces.

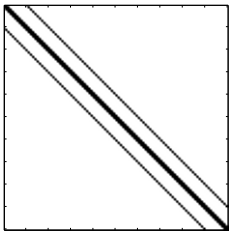
The answer to the partitioning question can depend subtly on the statistics of Z, and can most definitively be answered for the broad class of Markov random fields, discussed in Chapter 6.

A number of partitioning schemes are examined in this book, in particular a partitioning into adjacent groups (e.g., the partitioning of an image into rows or columns) in the marching methods of Section 10.1, and the hierarchical partitioning into nested subsets in the multiscale methods of Section 10.3.

5.6.4 Banded Inverse-Covariances

Motivated by the sparse-matrix methods described in Section 5.3, it is tempting to consider representing covariance matrices by a sparse, banded representation. However, a b-banded covariance P implies that any state element is correlated with only

[4] The reader may wonder about stability requirements for A; however for a domain of finite size stability is not an issue.



(a) 5-Banded Matrix

(b) 5-Banded Matrix Inverse

Fig. 5.18. The inverse of a sparse matrix is normally dense, with enormously complicated structure.

$b-1$ other elements, and completely decorrelated with the entire remainder of the field. For example, a standard five-banded covariance for a two-dimensional random field would imply that any pixel is correlated only with its immediate four neighbours. Thus for any reasonably sized b, the spatial extent of correlation will be very modest, *much* shorter than would arise in realistic contexts. For those rare locally-correlated problems it is much simpler to use a local estimator, so banded-covariances are of little interest.

Furthermore, to take a given covariance and to *force* it to be banded, by setting all values outside of the bands to zero, almost certainly results in a matrix which fails to be positive-definite.

Banded *inverse* covariances are a completely different matter, however. In particular, because the inverse of a sparse matrix is normally dense, a sparse P^{-1} can imply a dense P — that is, with *all* elements correlated with one another, as is illustrated in Figure 5.18. Indeed, even a five-banded P^{-1} can model arbitrarily strong correlations between arbitrarily distant state elements. Additional bands do not necessarily lead to stronger correlations, rather to more interesting variations, patterns, and textures.

However, the benefits of sparse banded inverses go far beyond the convenience of simplified covariance storage.

Indeed, the singular *key* benefit of most[5] sparse inverses is that they imply a certain decoupling of the random process, meaning that the original field can, relatively easily, be divided into smaller pieces, a key desirable property when considering domain decomposition methods. This large, important class of statistical models is known as Markov random fields, the topic of Chapter 6.

[5] Not *all* sparse inverses have simple or useful decoupling properties. Normally the process is reversed: we specify the nature or degree of decoupling, which determines the class of sparse inverses to be considered.

Even outside of an explicit consideration of sparse inverse-covariances this class of models is significant. The recursive-dynamic models of Section 5.6.3 possess a Markov decomposition property, by construction, and therefore also have sparse inverses, although this inverse is normally not computed and stored explicitly. Similarly the deterministic models of Section 5.5 are characterized by sparse L or $L^T L$, which we understand to correspond with P^{-1}, and are therefore also Markov.

5.7 Model Determination

The problem of inferring a model is known as system identification [212], a complex task whose details and subtleties are outside the scope of this text.

In many cases, if the multidimensional measured system is a real, physical one, then the behaviour of the system will be described by a specific mathematical or physical formulation, whether an elliptical partial differential equation with boundary conditions, or a regular differential equation over time, which may then be discretized.

However, even for physical systems with a known mathematical model, there are circumstances in which the mathematical model fails to lead to a meaningful prior:

- The mathematical description may live at a scale far finer than the intended discretization of the problem.
- The dimensionality of the mathematics may be different from the intended estimation problem. For example, the three-dimensional behaviour of water in the ocean is governed by the Navier–Stokes equation, which is of limited use in forming a prior of the two-dimensional ocean surface.

In these and many other cases we will have training data, but not a physical basis for knowing a model, although our understanding of the problem may allow us to assert whether the model is stationary, periodic at the boundaries, or isotropic, for example.

Under *modelling* we normally understand the inference of a model with relatively *few* parameters for representation. A very large model is much more likely to lead to overfitting, although methods of cross-validation (Section 2.4.1) can be used to test for that. In practice, for large multidimensional problems we would prefer to learn a parametrized model, meaning a statistical model $\left(P(\underline{\theta}), C(\underline{\theta}), R(\underline{\theta})\right)$ with some number of unknown parameters $\underline{\theta}$.

There are variations in terms of what is known:

- Ground truth data $\check{Z}$ for the state Z, or
- Some number of measurements sets M,

and what is not known:

- The prior model P or regularization constraint L, and/or
- The forward problem C, and/or
- The statistics of the measurements R.

It is quite common for the physics of the problem to dictate C and R, leaving the prior model to be inferred. In many cases a Markov model is chosen, as just discussed in Section 5.6.4, and for which specialized methods of model inference are discussed in greater detail in Chapter 6, in Section 6.6.

Outside of the Markov case, we need to estimate from the given data the parameters $\underline{\theta}$, parameterizing a prior model

$$Z \sim \mathcal{N}\big(\underline{0}, P(\underline{\theta})\big) \qquad Z \sim \mathcal{N}\big(\underline{0}, \mathcal{P}(\underline{\theta})\big) \qquad \text{or} \qquad Z \sim L(\underline{\theta}) \tag{5.88}$$

and so on, as appropriate. For example, all of the analytical positive-definite forms in Section 5.6.1 are a function of a single scalar parameter θ. We rarely have a prior model for $\underline{\theta}$ itself, so the unknown parameters $\underline{\theta}$ are estimated using maximum likelihood (2.93):

$$\hat{\underline{\theta}} = \arg_\theta \max p(\check{Z}|\underline{\theta}) \qquad \text{or} \qquad \hat{\underline{\theta}} = \arg_\theta \max p(M|\underline{\theta}) \tag{5.89}$$

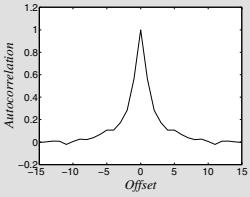
depending on whether ground truth $\check{Z}$ or measurements M are given [248].

In many cases, especially when measurements are given, it is exceptionally difficult to even write down $p(M|\underline{\theta})$, essentially because the relationship between $\underline{\theta}$ and M goes via (unknown) Z. In such cases the expectation-maximization algorithm [25, 85, 275], also discussed in Section 7.5, is widely used for parameter estimation.

Example 5.2: Model Inference

We are given a sample $\check{Z}$ of a random field Z and its associated correlation:

Sample

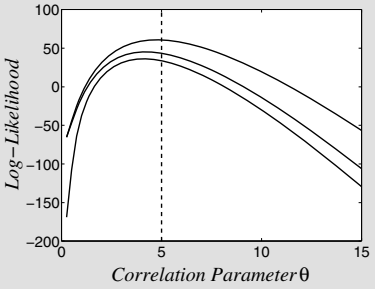


Correlation

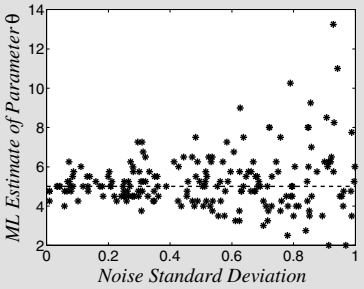
Example continues . . .

Example 5.2: Model Inference (cont'd)

In principle we could attempt to infer a model directly from the correlation, however the way in which the model parameters θ relate to the correlation may not be obvious. Suppose we know the form of the model, here the exponential correlation $f_{\text{Exp}}(r, \theta)$ of (5.61); then we can compute the likelihood $p(Z|\theta)|_{Z=\check{Z}}$ as a function of θ and select $\hat{\theta}$ to maximize:



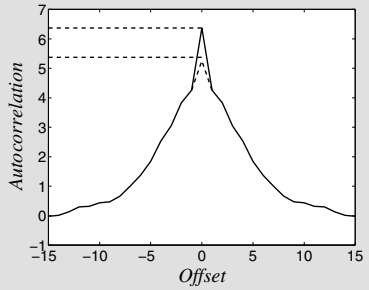
Log-Likelihoods for Three Samples (dashed line at true θ)



ML Estimates of θ from Noisy M (as a function of noise level)

Because the three samples $\check{Z}_i$ are different, the log-likelihood curves and their respective maxima will vary somewhat from sample to sample, left. If, furthermore, the sample $M = Z + V$ is noisy, right, then the accuracy of the estimate decreases at greater noise levels.

It is more likely that our model uncertainties would include correlation, variance, and measurement noise variance. These parameters can, in this simple case, be inferred from an autocorrelation:



By extrapolating a process variance of ≈ 5.1, the measurement noise variance appears as the increased height of ≈ 1.1 at zero-lag.

Instead, we could have estimated all three parameters in θ using maximum likelihood, an optimization over a three-dimensional space, for which an iterative method such as EM would typically be used.

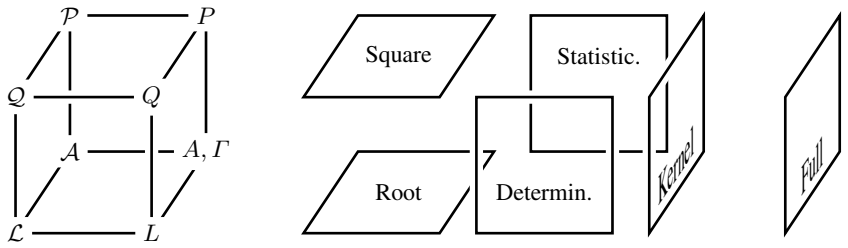


Fig. 5.19. We have seen eight forms of representation, left, where the forms differ based on the choice of three underlying aspects: representing a model or its square root, deterministic constraints versus statistical modelling, and a full/dense matrix versus a sparse matrix or convolutional kernel.

5.8 Choice of Representation

This chapter has developed a total of eight forms of representation: whether constraints or statistics, square root or squared, and whether a sparse matrix form or a stationary kernel, as illustrated in Figure 5.19.

The last of these three questions is perhaps of lesser significance, to the extent that kernels are just an implicit, highly sparse representation of a matrix. If the problem is highly nonstationary, then a regular sparse matrix approach is simpler. Alternatively, if the problem is mostly stationary, then it may be easier to store a kernel and to keep track of nonstationary exceptions (such as at boundaries and discontinuities).

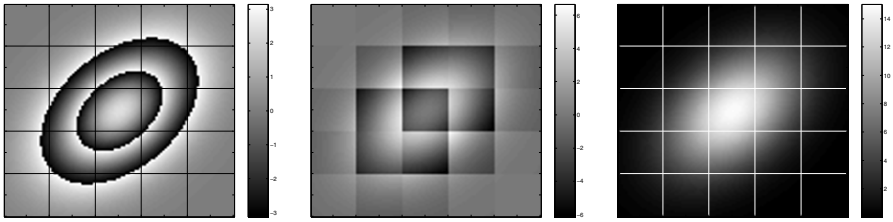
The former question is more significant. First, whereas *all* $L, \mathcal{L}, A, \Gamma$ are valid, the direct specification of $L^T L, \mathcal{Q}, P$ must guarantee positive-semidefiniteness. Although this requirement is easily satisfied for stationary problems, for nonstationary problems definiteness may be much more subtle.

In general a covariance P will be dense, and the detailed discussion of sparse statistical models is deferred to Chapter 6. For a constraints-based model, if the number of diagonal bands in L, or equivalently the number of nonzero entries in $\mathcal{L}$, is b, then $Q = L^T L$ may have up to b^2 bands. Therefore for $\underline{z} \in \mathbb{R}^n$, the product $\underline{x} = Q\underline{z}$ has complexity $\mathcal{O}(b^2 n)$, whereas separately computing

$$\underline{y} = L\underline{z}, \qquad \underline{x} = L^T \underline{y} \tag{5.90}$$

has considerably reduced complexity $\mathcal{O}(2bn)$, but at double the storage complexity, since $\underline{y}$ needs to be stored in memory.

In those cases where the computational complexity is not prohibitive, the reduced storage demands and the simpler, more homogeneous structure of Q versus the



Sample Interferogram with Regions | Each Region Individually Unwrapped | Regions Unwrapped Against Each Other

Fig. 5.20. A domain-decomposition approach to phase unwrapping [53]: Given an interferogram (left) we can divide it into regions and unwrap each region individually (center), where then the regions can be unwrapped against each other (right) to form the final image.

stacked sets of constraints in L (as in (5.47)) may lead to a preference for $Q, \mathcal{Q}$ over $L, \mathcal{L}$.

Application 5: Synthetic Aperture Radar Interferometry [53]

A fascinating problem in image processing and remote sensing is that of radar interferometry [134, 138].

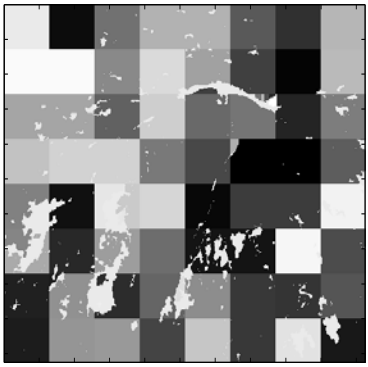
The complex radar return at a pixel has a phase proportional to the target distance at that pixel. It is possible to take *two* radar images, either from one satellite (e.g., Radarsat) from two precisely-defined orbital positions or, preferably, from a pair of satellites (e.g., ERS-1/2) having a precise orbital offset. It is possible to design the orbital positions of the imaging satellites such that the complex phase *difference* between the two images is proportional to height, allowing detailed topographic maps to be made from space. An example of a resulting phase-difference image, known as an interferogram, is shown in Figure 5.20.

We would like to recover the absolute phase difference ϕ, however the measured phase difference $m = c(\phi)$ is wrapped, meaning that $-\pi \leq c(\phi) < \pi$. The solution to the unwrapping problem is therefore some integer multiple of 2π:

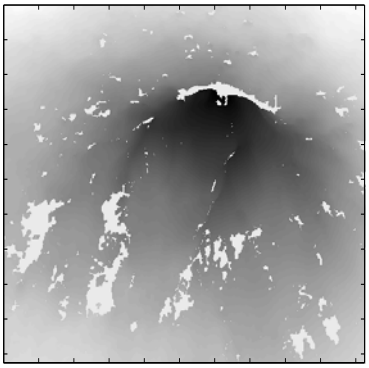
$$\phi = c(\phi) + 2\pi z. \tag{5.91}$$

It is possible to estimate z by counting the interferometric fringes, which are quite obvious in Figure 5.20, however real data are subject to noise and poor radar coherence, in which case the fringes are far less well defined. Integer (network-flow)

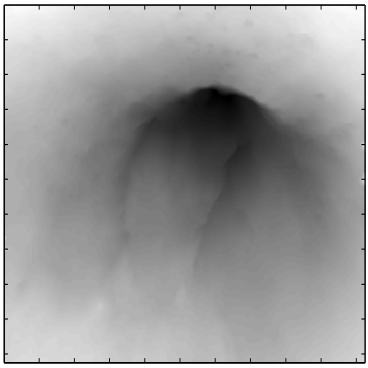
Mount Etna Interferogram



Division Into 80 Regions



Region-Based Reconstruction



Reconstruction Plus Interpolation

Fig. 5.21. The domain-decomposition phase unwrapping from Figure 5.20 applied to ERS-1/2 interferometric data from Mt. Etna [53]. The phase data are ignored in areas with low radar coherence, leading to the gaps in the regions and reconstruction images, and only those gaps are interpolated in the final image.

optimization methods have been used with considerable success [72], however there are issues of computational complexity in solving problems on a very large domain.

Motivated by the domain-decomposition ideas illustrated in Figures 5.2 and 5.5, Figure 5.20 shows the original interferogram being divided [53] into modestly-sized regions, with each region unwrapped individually, followed by unwrapping the regions against each other.

The approach has been applied to large problems and real data, one example of which is illustrated in Figure 5.21 for ERS data.

For Further Study

There is relatively little material on the subject of multidimensional modelling; indeed, it is precisely this gap which motivated the writing of this text.

Two classic papers by Terzopoulos [303, 304] examine methods for solving inverse problems given stationary kernels, with exceptions for folds and discontinuities, very much along the lines of aspects of this chapter. A related paper is that by Szeliski [299], although that paper is more easily read after Chapter 8.

Sample Problems

Problem 5.1: Membrane Kernel

Consider a 5×5 random field Z. Create the first-order, free-boundary 20×25 constraint matrices L_x, L_y, leading to the overall first-order constraint

$$L_1 = \begin{bmatrix} L_x \\ L_y \end{bmatrix}$$

Compute the squared constraints matrix $Q = L_1^T L_1$, and lexicographically unwrap the middle row of Q to see the membrane kernel; that is, compute

$$[Q_{13,1\ldots25}]_{5\times5}$$

Try unwrapping other rows of Q, such as the first and the last, and comment on what you observe.

Problem 5.2: Membrane Kernel

Repeat Problem 5.1 for the second-order thin-plate case. That is, create a constraints matrix

$$L_2 = \begin{bmatrix} L_{xx} \\ L_{yy} \\ 2L_{xy} \end{bmatrix}$$

and compute $Q = L_2^T L_2$. Unwrap the middle row of Q, along with a few other rows, and state your observations.

Problem 5.3: Matrix Truncation

In most circumstances, taking a covariance P and keeping only a certain number of diagonal bands leaves you with a matrix which is not positive-definite, and which is therefore invalid as an approximate prior model for estimation. Let

$$P^b = \text{bands}(P, b)$$

be a matrix which copies the b bands on either side of the principal diagonal, and sets the remaining entries in P^b to zero.

The positive-definiteness of P^b can be determined from its eigendecomposition; investigate the positive-definiteness of $P^b(\sigma)$ as a function of b and σ for two different covariances:

(a) **Gaussian:** Let P be a 20×20 covariance matrix, corresponding to a one-dimensional process of length twenty, such that

$$P_{i,j}(\sigma) = \exp\left(-\frac{(i-j)^2}{2\sigma^2}\right) \qquad P^b_{i,j}(\sigma) = \begin{cases} \exp\left(-\frac{(i-j)^2}{2\sigma^2}\right) & |i-j| \le b \\ 0 & |i-j| > b \end{cases}$$

(b) **Exponential:** Let P be a 20×20 covariance matrix, corresponding to a one-dimensional process of length twenty, such that

$$P_{i,j}(\sigma) = \exp\left(-\frac{|i-j|}{\sigma}\right) \qquad P^b_{i,j}(\sigma) = \begin{cases} \exp\left(-\frac{|i-j|}{\sigma}\right) & |i-j| \le b \\ 0 & |i-j| > b \end{cases}$$

A look at Problem 2.4 may help to explain what is happening here.

Problem 5.4: Two-Dimensional Interpolation

Suppose we have a 20×20 two-dimensional grid Z. Suppose also that we have seven measurements of individual points $z_{i,j}$:

i	5	15	5	15	3	10	17
j	2	2	10	10	18	17	18
m	15	0	10	5	20	10	20

Thus in the measurement model $\underline{m} = C\underline{z} + \underline{v}$, matrix C is a 7×400 array. The measurement errors are independent and have a variance of 9, thus

$$\text{cov}(\underline{v}) = R = 9 \cdot I.$$

We define four cases of regularizing constraints:

1. First-order constraints along i and j, free boundaries
2. Second-order constraints along i and j, periodic boundaries
3. First-order along i, second-order along j, a discontinuity (cut) from (10.5,1) to (10.5,10), periodic boundaries at $i = 1, 20$, free boundaries at $j = 1, 20$
4. Second-order along i and j, a fold from (10,10) to (10,20), periodic boundaries at $j = 1, 20$, free boundary at $i = 1$, zero boundary at $i = 20$

For each of these four cases, do the following:

(a) Compute the constraints matrix L and then compute $Q = L^T L$. Plot the structure of Q (using `spy(Q)` in MATLAB), and interpret.
(b) Choose one or two rows from Q and lexicographically unwrap them to see the associated kernels; interpret the results.
(c) Select an appropriate value of λ (by trial-and-error or by cross-validation), compute estimates $\hat{Z}$, and state your observations.

Problem 5.5: Open-Ended Problem — Nonstationary Modelling

Suppose we wish to consider a global remote-sensing problem, such as the Geoid of Figure 5.12, the altimetric problem of Figure 1.3, or the ocean-temperature problem of Figure 4.1 and Application 8.

The problem is very large-scale, nonstationary, with periodic boundaries. We wish to construct a prior model subject to the following:

1. We wish to model the Earth on a rectangular lattice, gridded in latitude and longitude, with lattice points one degree apart, with a latitude range from 60S to 60N.

2. The smoothness constraint is second order over water, state elements over land should be removed.

3. The west–east boundaries need to be periodic; the north–south boundaries should be free.

4. The spatial correlation length will be stationary, however because the lattice is gridded in lat–long, and lines of latitude converge at the poles, therefore the lattice model is *non*stationary. Specifically, the L_{yy} terms are stationary (lines of longitude are parallel), however the L_{xx} terms need to be scaled.

5. Clearly, the entire model needs to be positive-definite.

Because the model is large and nonstationary, we want to specify the constraints L, and not the weight matrix Q.

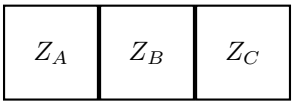
(a) Draw a sketch or explain in simple terms how to go about formulating L.
(b) Implement code to generate the product $L\underline{z}$.
(c) Optional: for those readers who have read or are familiar with the linear-systems and preconditioning methods of Chapter 9, generate estimates based on L given real or artificial data.

6

Markov Random Fields

Based on the modelling discussions of Chapter 5, the issues of computational and storage complexity for large problems have motivated an interest in sparse representations, and also in those models which allow some sort of decoupling, or domain decomposition, to allow a hierarchical approach. As we shall see, both sparsity and domain decomposition are at the heart of all *Markov* processes, thus the topic of Markovianity is central to the modelling and processing on large domains.

Fundamentally, all things Markov [127, 276] have a conditional independence or conditional decorrelation property. For example, given three pieces of a random field,



(6.1)

then B is a Markov separation of A and C if it contains all of the information needed by A about C:

$$p(Z_A|Z_B, Z_C) = p(Z_A|Z_B) \qquad p(Z_C|Z_B, Z_A) = p(Z_C|Z_B). \tag{6.2}$$

That is, given B, knowing C tells us nothing *more* about A, nor knowing A any more about C. In other words, having determined B (somehow), we are free to consider A and C separately: the problem has been decoupled!

This Markov property is a common phenomenon in familiar random processes, it conveniently extends to two- and higher-dimensional fields, and it leads to rather flexible models and algorithms.

This chapter summarizes the properties of Markov models, first in one dimension and then in higher-dimensional cases, and discusses methods of Markov model inference. How to use the Markov decoupling properties and how to compute estimates based on Markov priors is the subject of Chapters 7 through 10.

P. Fieguth, *Statistical Image Processing and Multidimensional Modeling*, Information Science and Statistics, DOI 10.1007/978-1-4419-7294-1_6, © Springer Science+Business Media, LLC 2011

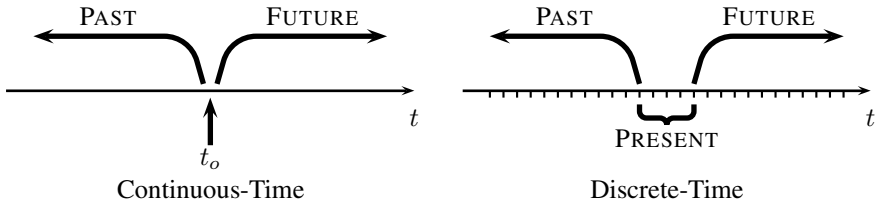


Fig. 6.1. The past and the future of one-dimensional Markov processes are conditionally separated by some information of the present. In continuous time, the required information is the state value, and some number of derivatives, at a single point in time t_o. In discrete time, the "present" consists of some number of consecutive samples.

6.1 One-Dimensional Markovianity

A one-dimensional random process $z(t)$ is Markov (Figure 6.1) if the knowledge of the process at some point z_o decouples the "past" z_p and the "future" z_f:

$$p(z_f|z_o, z_p) = p(z_f|z_o), \qquad p(z_p|z_o, z_f) = p(z_p|z_o). \tag{6.3}$$

The specifics of what these terms really mean are dependent on the order of the process, and whether we are working with continuous-time or discrete-time processes. In continuous time, if a process is decoupled about a given point t_o

$$\begin{aligned} z_p &= \big\{z(t)|t < t_o\big\} \\ z_f &= \big\{z(t)|t > t_o\big\}, \end{aligned} \tag{6.4}$$

where the information kept at t_o is the process and its first $(n-1)$ derivatives,

$$z_o = \left\{z(t_o), z^{(1)}(t_o), \ldots, z^{(n-1)}(t_o)\right\}, \tag{6.5}$$

then z is said to be nth-order Markov. Similarly, in the discrete-time case, n successive process samples are needed to separate an nth-order Markov process:

$$\begin{aligned} z_p &= \big\{z(t)|t \leq t_o - n\big\} \\ z_f &= \big\{z(t)|t > t_o\big\} \\ z_o &= \big\{z(t_o - n + 1), z(t_o - n + 2), \ldots, z(t_o)\big\}. \end{aligned} \tag{6.6}$$

Clearly these two definitions for continuous-time and discrete-time are mutually compatible as the discretization interval goes to zero, since the first $n-1$ synthetic derivatives of a process can be computed from n successive samples.

One-dimensional Markov processes are generally of limited interest in solving multidimensional problems, so we focus on the generalization of these ideas to higher dimensions in Section 6.2. There are, however, two well-known classes of one-dimensional Markov processes, discussed below.

6.1.1 Markov Chains

The special case of first-order, one-dimensional, discrete-time, discrete-state Markov processes is also known by the more familiar name of Markov chains, which were introduced in Section 4.5.1.

The first-order and discrete-time nature of the process implies that

$$p\big(z(t+1)|z(t), z(t-1), z(t-2), \ldots\big) = p\big(z(t+1)|z(t)\big) \tag{6.7}$$

and the discrete-state nature of the process implies that, rather than a continuous probability density, we have a discrete set of state-to-state transition probabilities

$$a_{ij} = \Pr\big(z(t+1) = s_i \in \Psi | z(t) = s_j \in \Psi\big). \tag{6.8}$$

Although Markov chains provide a helpful intuitive connection, in practice they contribute relatively little to image modelling and estimation:

- The number of states tends to be impossibly large, in practice. If the Markov chain represents a set of n eight-bit grey-level pixels, then the number of states is 256^n.
- The one-dimensionality and causality of Markov chains is too limiting to apply to multidimensional contexts.

We do, however, return to Markov chains in looking at hidden Markov models in Chapter 7.

6.1.2 Gauss–Markov Processes

A random process is nth-order Gauss–Markov if it is nth-order Markov and if the elements of the process are jointly Gaussian. As an example of key importance, our canonical dynamic model (4.1) of Section 4.1 is, by construction, jointly Gaussian (based on its linearity and Gaussian driving noise) and, in fact, is also first-order Markov. Indeed, Markovianity is explicit in its recursive form:

$$\underline{z}(t+1) = A(t)\underline{z}(t) + B(t)\underline{w}(t) \qquad \underline{w}(t) \sim \mathcal{N}(\underline{0}, I), \tag{6.9}$$

which implies that given $\underline{z}(t)$, the uncertainty remaining in $\underline{z}(t+1)$ is due entirely to $\underline{w}(t)$. The noise term $\underline{w}(t)$ is uncorrelated with $\underline{z}(t-1), \underline{z}(t-2), \ldots$ therefore none of $\underline{z}(t-1), \underline{z}(t-2), \ldots$ have any additional information regarding $\underline{z}(t+1)$ and thus the conditional probability density decomposes as

$$p\big(\underline{z}(t+1)|\underline{z}(t), \underline{z}(t-1), \ldots\big) = p\big(\underline{z}(t+1)|\underline{z}(t)\big). \tag{6.10}$$

By extension, all such recursive-Gaussian models are Gauss–Markov, examples of which are listed below and plotted in Figure 6.2:

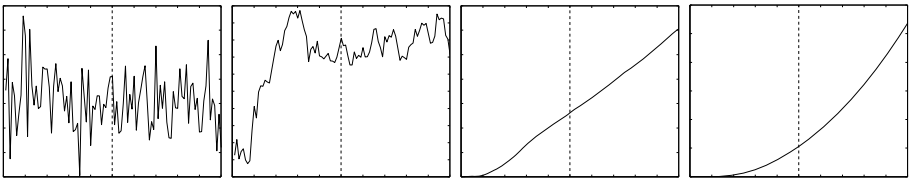


Fig. 6.2. The order of a Markov process counts the number of degrees of freedom (process value plus derivatives) needed at some point, such as the dashed line, to characterize the process at that point. The order is normally related to process smoothness and complexity.

$$\begin{array}{lll} \text{White noise} & x(t) = w(t) & \text{0th-order Gauss–Markov} \\ \text{Brownian motion} & x(t) = x(t-1) + w(t) & \text{1st-order Gauss–Markov} \\ \text{Autoregressive} & x(t) = \sum_{i=1}^{n} \alpha(i) x(t-i) + w(t) & n\text{th-order Gauss–Markov} \end{array} \tag{6.11}$$

With the connection established between Markovianity and linear dynamic models, the discussion of Section 4.1.1 becomes germane here. In particular, we saw that an nth-order scalar autoregressive or nth-order scalar moving average process, both of which are nth-order Markov could, via state augmentation, be rewritten in first-order n-vector form (6.9). Thus we understand that the notion of Markov order is somewhat fluid, and furthermore that the modelling of higher-order Markov processes is not necessarily inherently difficult, but may instead just require an increase in state dimension.

6.2 Multidimensional Markovianity

The generalization of Markov decoupling to two and higher dimensions is known as a *Markov Random Field* [62, 207, 335]. The immediate difficulty is that the one-dimensional concepts of "past" and "future" are lost, inasmuch as there is no natural ordering of the elements in a multidimensional grid. Instead, a random field $\underline{z}$ on a lattice (or grid) Ω is Markov (Figure 6.3) if the knowledge of the process on a boundary set b decouples the inside and outside of the set:

$$p(\underline{z}_i | \underline{z}_b, \underline{z}_o) = p(\underline{z}_i | \underline{z}_b), \qquad p(\underline{z}_o | \underline{z}_b, \underline{z}_i) = p(\underline{z}_o | \underline{z}_b). \tag{6.12}$$

This boundary concept, although elegantly intuitive, is lacking in details: how "thick" does the boundary need to be? Five examples of boundaries, with varying degrees of ambiguity, are shown in Figure 6.4.

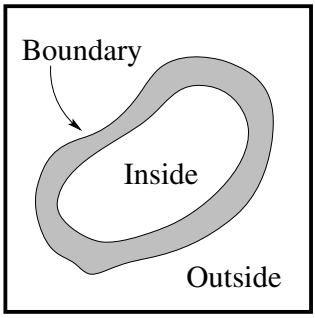


Fig. 6.3. The notion of Markovianity extended to higher dimensions: A multidimensional random process is Markov, with respect to some boundary, if the process within the boundary (shaded) decouples the process inside and outside.

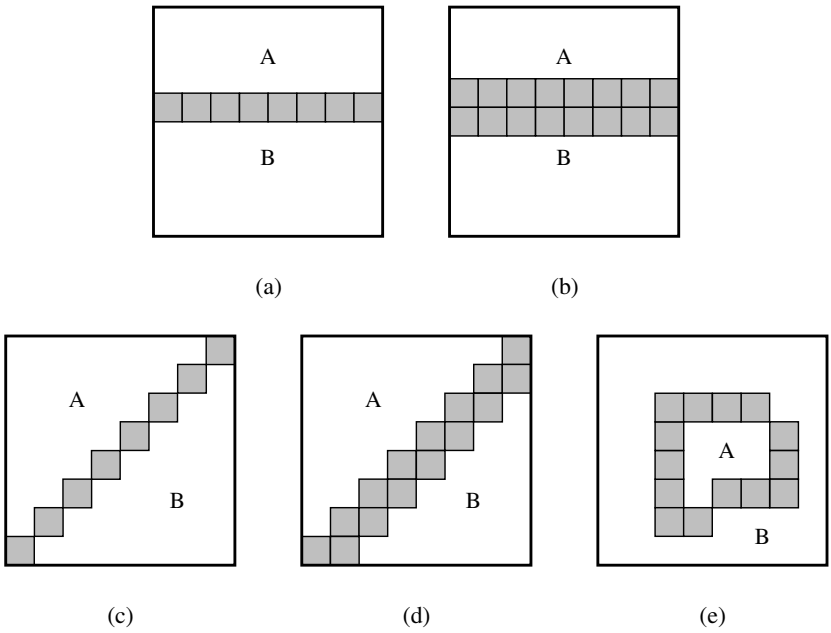


Fig. 6.4. The "thickness" of the boundary is clear enough in cases (a) and (b), but how do we define the thickness of a diagonal boundary (c,d) or in more heterogeneous cases (e)?

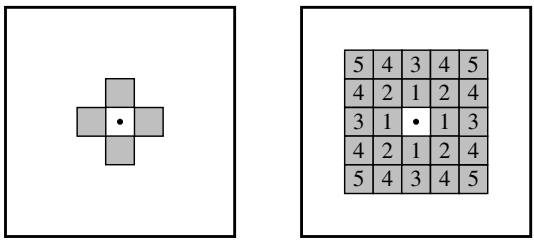


Fig. 6.5. (a) The simplest neighbourhood structure for a two-dimensional Markov process: Any pixel (black dot) is decoupled from the rest of the domain when conditioned on its four immediate (shaded) neighbours. (b) The definition of two-dimensional neighbourhood order: for an nth-order model, the neighbourhood conditionally separating the shaded region from the rest of the domain contains the labeled pixels from one to n. The neighbourhood in (a) is therefore first order.

It is often simpler, and more explicit, to talk about decorrelating or decoupling a *single* lattice site j from the rest of the lattice $\Omega\backslash j$ by conditioning on a local neighborhood $\mathcal{N}_j$ [86, 127]:

$$p\big(z_j|\{z_k, k \in \Omega\backslash j\}\big) = p\big(z_j|\{z_k, k \in \mathcal{N}_j\}\big), \tag{6.13}$$

as illustrated in Figure 6.5. For a neighbourhood to be valid it must satisfy two simple requirements:

1. A site is not its own neighbour: $j \notin \mathcal{N}_j$.
2. The neighbourhood property must reciprocate:

$$j \in \mathcal{N}_k \;\Leftrightarrow\; k \in \mathcal{N}_j, \tag{6.14}$$

   from which it is clear that the random field is *loopy*, in the sense of Figure 5.1, which will lead to challenges in computation.

In most circumstances the shape of neighbourhood $\mathcal{N}_j$ is not a function of location j. For the neighbourhood structure to be stationary, the reciprocating property demands that the neighbourhood be symmetric:

$$j + \delta \in \mathcal{N}_j \;\Rightarrow\; j - \delta \in \mathcal{N}_j \tag{6.15}$$

for any offset δ, a property which holds true for common neighbourhood structures. The shape and extent of $\mathcal{N}_j$, measured by the *order* of the neighbourhood, is one of

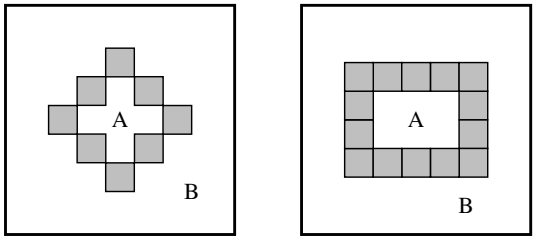


Fig. 6.6. From Figure 6.5 it follows that the left and right panels show first- and second-order boundaries, respectively.

the fundamental properties characterizing a random field; a first-order neighborhood is shown in Figure 6.5, along with the numbering scheme for higher-order neighborhoods. It is crucial to understand that the size of image structure which can be realized by a Markov model is *not* related to model order or neighbourhood size — a first-order field could exhibit long correlations, and a fourth-order field possibly short ones — however the *complexity* of the realized structure increases with order. Thus the first-order neighbourhood of Figure 6.5(a) can lead to models in which widely-separated pixels are highly correlated (what we mean by a "large" structure), but it is limited to only very simple patterns or textures.

The above discussion is entirely formulated in terms of the probability density $p(\underline{z})$, which is impractical for large random fields; the following sections summarize the two broad alternatives which have been developed:

1. Gauss–Markov Random Fields: $\underline{z}$ is Gaussian, in which case the field can be characterized explicitly in terms of expectations rather than probability densities.
2. Gibbs Random Fields: an energy $H(\underline{z})$ is associated with each possible field $\underline{z}$; a probability density is then constructed implicitly from $H(\underline{z})$ in such a way that $p(\underline{z})$ satisfies (6.12), (6.13).

6.3 Gauss–Markov Random Fields

So far, all notions of domain decomposition have been written in terms of probability distributions and conditional independence, which are very inconvenient numerically. Instead, a related notion is that of conditional decorrelation, which implies the absence of any *linear* relationship, as opposed to conditional independence, which

implies the absence of *any* relationship. In the Gaussian case [61], meaning that $\underline{z}$ is jointly Gaussian, decorrelatedness and independence are equivalent. Because correlations are much simpler than probability densities, we are motivated to consider this more limited notion of Markovianity.

Given a neighbourhood structure $\{\mathcal{N}_j\}$, we say that a random process z is *Strongly Markov* [86] with respect to $\{\mathcal{N}_j\}$ if z is conditionally independent as

$$p\big(z_j|\{z_k, k \in \Omega\backslash j\}\big) = p\big(z_j|\{z_k, k \in \mathcal{N}_j\}\big). \tag{6.16}$$

This is the definition of Markovianity, copied from (6.13), that we have used up to this point.

Next, we say that a random process z is *Weakly Markov* [86] with respect to $\{\mathcal{N}_j\}$ if z conditionally decorrelates as

$$E\big[z_j|\{z_k, k \in \Omega\backslash j\}\big] = E\big[z_j|\{z_k, k \in \mathcal{N}_j\}\big], \tag{6.17}$$

in contrast to (6.13). We recognize such a conditional expectation as the Bayesian estimate (3.46) of z_j, which is linear in the Gaussian case, but nonlinear in general.

Formulating a nonlinear Bayesian estimate is really not of interest here, rather we seek a simple alternative to Markov fields based purely on second-order statistics. Therefore the much simpler, and more widely used, definition is that of a *Wide-Sense Markov* random field [86] with respect to $\{\mathcal{N}_j\}$:

$$\hat{z}_j = E_{\text{LLSE}}\big[z_j|\{z_k, k \in \Omega\backslash j\}\big] = E_{\text{LLSE}}\big[z_j|\{z_k, k \in \mathcal{N}_j\}\big], \tag{6.18}$$

where E_{LLSE} explicitly refers to a *linear* expectation, the best linear estimate of z_j, as discussed in Section 3.2. In the Gaussian case Markov, weakly Markov, and wide-sense Markov are all equivalent.

It is also true that a zero-mean random process z is wide sense Markov with respect to $\{\mathcal{N}_j\}$ if

$$z_j = \sum_{k \in \mathcal{N}_j} \bar{g}_{j,k} z_k + w_j \qquad E[z_k w_j] = 0 \;\; \forall j \neq k. \tag{6.19}$$

The equivalence of these two forms is easily seen:

(6.18) $\Longrightarrow$ (6.19): E_{LLSE} is a linear estimator for z_j. From the left-hand side of (6.18) this estimator is global and is therefore the optimum linear estimator. Therefore by orthogonality the estimation error w_j will be uncorrelated with all coefficients z_k. The right-hand side of (6.18) asserts that the estimator can be written locally, within the neighbourhood, as in (6.19).

(6.19) $\Longrightarrow$ (6.18): The expression (6.19) defines a linear estimator

$$\hat{z}_j = \sum_{k \in \mathcal{N}_j} \bar{g}_{j,k} z_k, \tag{6.20}$$

having an estimation error

$$\tilde{z}_j = \hat{z}_j - z_j = -w_j \tag{6.21}$$

meaning that w_j obeys orthogonality. By definition this must therefore be the linear least-squares estimator (LLSE), implying (6.18).

This equivalence is particularly useful, because it relates a conceptual notion of decoupling (6.18) with an associated linear estimator in autoregressive form (6.19).

It is particularly illuminating to derive the statistics of w and z from (6.19). First, rewrite (6.19) as

$$\sum_k g_{j,k} z_k = w_j \qquad g_{j,k} = \begin{cases} -\bar{g}_{j,k} & k \in \mathcal{N}_j \\ 1 & j = k \\ 0 & \text{otherwise} \end{cases} \tag{6.22}$$

If we let the noise variance $\text{var}(w_j) = \sigma_j^2$, then for two sites $j \neq k$,

$$E[w_j w_k] = E\left[w_j \sum_i g_{k,i} z_i \right] \tag{6.23}$$

$$= g_{k,j} E[w_j z_j] = g_{k,j} E[w_j w_j] = g_{k,j} \sigma_j^2 \tag{6.24}$$

from which the reciprocity requirement (6.14) on the neighbourhood follows:

$$E[w_j w_k] = E[w_k w_j] \implies g_{k,j} \sigma_j^2 = g_{j,k} \sigma_k^2. \tag{6.25}$$

Thus we see that the noise process w is *not*, in fact, white: there are off-diagonal correlations. Indeed, it has a correlation structure given by the linear estimation coefficients g, where the sparseness and locality of the correlation structure are exactly those of the random field neighbourhoods.

If we lexicographically reorder z, w into vectors, as usual, then

$$\begin{aligned} (6.22) &\implies G\underline{z} = \underline{w} \\ (6.24) &\implies \text{cov}(\underline{w}) = G\Sigma\,, \end{aligned}$$

where G collects the estimator coefficients $\{g_{i,j}\}$, and $\Sigma_w = \text{Diag}(\sigma_1^2, \sigma_2^2, \ldots)$ is a diagonal matrix with the noise variances σ_j^2 along the diagonal. Although $G\Sigma$ gives the appearance of being asymmetric, the constraints relating g and σ^2 in (6.25) do ensure the required symmetry of $\text{cov}(\underline{w})$.

Because $G\Sigma$ is the covariance of a random vector it must (for any valid model) be invertible, from which it follows that G is invertible, thus

$$\underline{z} = G^{-1}\underline{w} \quad \implies \quad \text{cov}(\underline{z}) = G^{-1} G\Sigma G^{-T} = \Sigma G^{-T}. \tag{6.26}$$

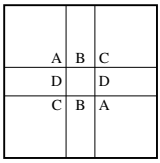


Fig. 6.7. The symmetry structure required by the LLSE kernel of a stationary Gauss–Markov random field.

The most profound result, however, follows from the *inverse* covariance of the random field,

$$\big(\mathrm{cov}(\underline{z})\big)^{-1} = G^T \Sigma^{-1}, \tag{6.27}$$

leading to the following significant conclusions:

1. Whereas the entries in a covariance describe the correlation between two random-field elements, the entries in a covariance *inverse* correspond to the associated linear estimator. That is, the covariance-inverse describes a *model*.
2. The sparsity and locality of a matrix inverse are directly related to the neighbourhood structure of the associated random field.
3. Assuming a covariance inverse to be sparse-banded is equivalent to asserting that the corresponding random field is wide-sense Markov.

The study of Markov random fields has thus led to a much deeper understanding of inverse-covariances, in particular the meaning of sparsity.

Finally, we understand that a given Markov model g or G implicitly specifies the random-field prior $P^{-1} = G^T \Sigma^{-1}$, thus random-field estimation proceeds as usual:

$$\hat{\underline{z}} = \big(C^T R^{-1} C + P^{-1}\big)^{-1} C^T R^{-1} \underline{m} \tag{6.28}$$

$$= \big(C^T R^{-1} C + G^T \Sigma^{-1}\big)^{-1} C^T R^{-1} \underline{m}. \tag{6.29}$$

To take advantage of the sparsity normally encountered in C, R and G, this estimator would be rewritten to remove the matrix inversion as

$$\big(C^T R^{-1} C + G^T \Sigma^{-1}\big)\, \hat{\underline{z}} = C^T R^{-1} \underline{m}. \tag{6.30}$$

A very important special case of Gauss–Markov random fields is *stationarity*, in which case the local estimator g and the driving noise statistics σ are both location invariant:

$$g_{j,k} = g_{k-j} \qquad \sigma_j^2 = \sigma^2 \;\Rightarrow\; \Sigma = \sigma^2 I, \tag{6.31}$$

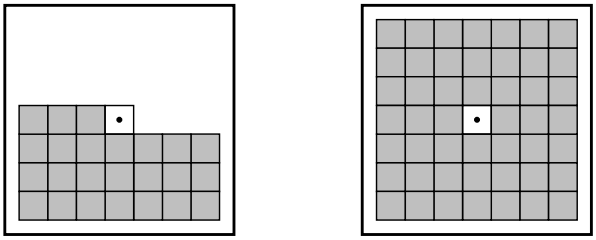


Fig. 6.8. Regions of support for causal (left) and noncausal (right) neighborhoods of the central element. The half plane illustrated on the left is only one of many possible choices.

so that the random field statistics are

$$\text{cov}(\underline{w}) = G\sigma^2 \qquad \text{cov}(\underline{z})\; G^{-1}\sigma^2 \qquad E\left[\underline{z}\,\underline{w}^T\right] = I\sigma^2. \tag{6.32}$$

Combining location invariance with the reciprocity symmetry of (6.25) leads to the constrained estimation weights

$$g_{j-k} = g_{k-j}, \tag{6.33}$$

as sketched in Figure 6.7, a constraint which essentially reflects the fact that covariance matrices are symmetric. The random field stationarity then allows the prior model to be expressed in an efficient convolutional kernel form, in which case the prior term of (6.30) becomes

$$P^{-1}\underline{\hat{z}} = G^T\Sigma^{-1}\underline{\hat{z}} = \sigma^{-2}G\,\underline{\hat{z}} = \sigma^{-2}\left[\mathcal{G} * \hat{Z}\right]_{:} \tag{6.34}$$

The two key remaining questions, then, are how to infer the model parameters g, addressed in Section 6.6, and how to efficiently estimate and sample from a given model, addressed in Chapters 9 and 11.

6.4 Causal Gauss–Markov Random Fields

If a random field $\underline{z}$ is causal, each neighborhood $\mathcal{N}_j$ must limit its support to one half of the plane, as sketched in Figure 6.8. These are known as nonsymmetric half-plane (NSHP) models [7], and lead to very simple autoregressive equations for sampling and estimation.

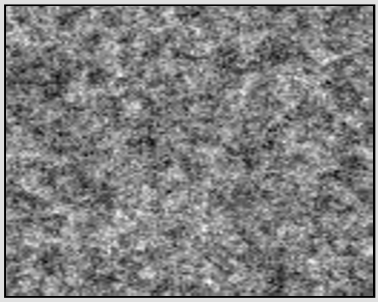
Specifically, there must exist an ordering of the field elements, a *sequential* model in the sense of Figure 5.1, such that each element depends only on the values of elements lying earlier in the ordering; let

Example 6.1: Example Markov Kernels

We have seen that there is a relationship between an inverse-covariance P^{-1} and MRF coefficients, and also between P^{-1} and constraints Q. We can therefore consider the Membrane or Thin-Plate kernels of Figure 5.14 as MRF models.

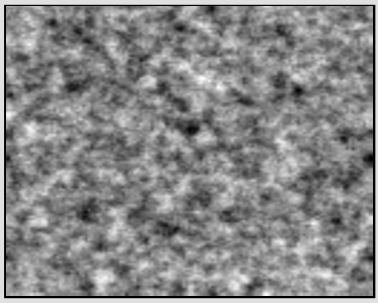
$$\begin{array}{ccc} & -1 & \\ -1 & \boxed{4.1} & -1 \\ & -1 & \end{array}$$

Membrane



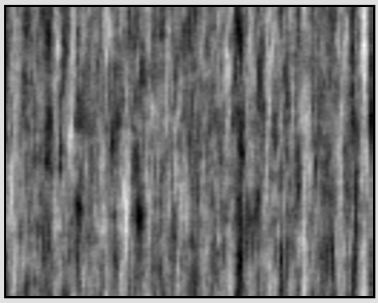
$$\begin{array}{ccccc} & & 1 & & \\ & 2 & -8 & 2 & \\ 1 & -8 & \boxed{20.1} & -8 & 1 \\ & 2 & -8 & 2 & \\ & & 1 & & \end{array}$$

Thin-Plate



$$\begin{array}{ccccc} & -91 & 517 & 8 & \\ 58 & 1405 & -5508 & 1164 & 85 \\ -139 & -2498 & \boxed{10000} & -2498 & -139 \\ 85 & 1164 & -5508 & 1405 & 58 \\ & 8 & 517 & -91 & \end{array}$$

"Tree-Bark" [195]



Just as the correlation length or feature size of the membrane and thin-plate kernels could be varied (Figure 5.15), similarly the feature size in the MRF samples is not a function of neighbourhood size, but rather of the parameters in the model. Instead, as the neighbourhood size increases (from top to bottom) it is the *complexity* of the structure in the random field which increases, not its scale.

All three sampling examples were generated using the FFT method discussed in Section 8.3.

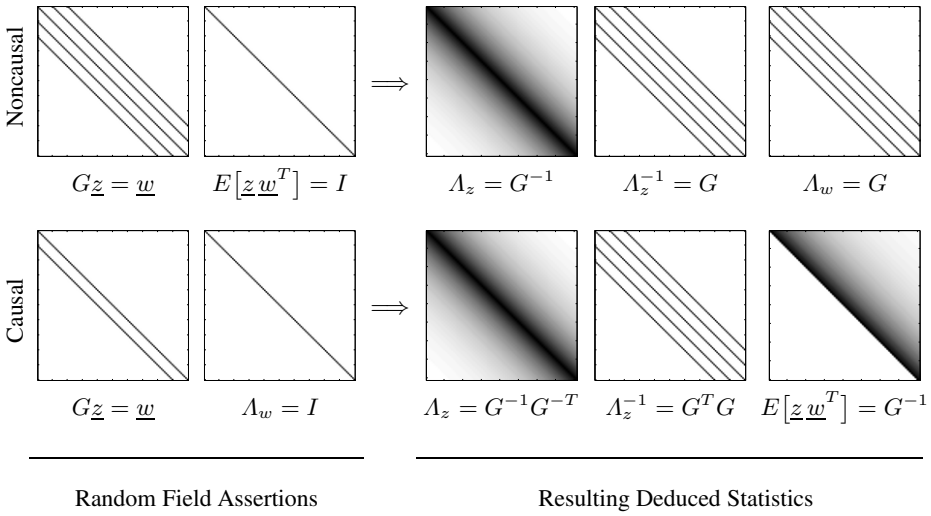


Fig. 6.9. A broad overview comparing the assumptions and statistics of noncausal (top) and causal (bottom) random fields, assuming the noise variance $\sigma^2 = 1$. The random field statistics, right, can be deduced or computed from the model assertions, left. In the noncausal case, we assert the local model G and the decorrelatedness of field and process noise, whereas in the casual case we assert the local model G and the whiteness of the process noise. In both cases the inverse-covariance of the random field is sparse.

$$\mathcal{N} < j \tag{6.35}$$

imply that all of the elements of $\mathcal{N}$ appear earlier in the ordering than j, that is, in the half-plane preceding j. Then, in parallel with the definition (6.18), (6.19) for the noncausal case, a random process z is causally wide-sense Markov with respect to a causal neighbourhood system

$$\mathcal{N}_j < j \;\; \forall j \tag{6.36}$$

if

$$\hat{z}_j = E_{\text{LLSE}}\big[z_j | \{z_k, k < j\}\big] = E_{\text{LLSE}}\big[z_j | \{z_k, k \in \mathcal{N}_j\}\big], \tag{6.37}$$

or, equivalently, if

$$z_j = \sum_{k \in \mathcal{N}_j} g_{j,k} z_k + w_j, \qquad E[z_k w_j] = 0 \; \forall k < j, \tag{6.38}$$

where the noise w_j in the autocorrelation (6.38) can now be chosen to be white. It is important to note the three differences between the formulation of the causal and noncausal cases, illustrated visually in Figure 6.9:

1. The neighbourhood structure is causal, $\mathcal{N}_j < j$, therefore reciprocity no longer applies.

2. The Markov decoupling in (6.37) is with respect to elements in the half plane, not with the entire domain.

3. The estimation error w in (6.38) is orthogonal to elements in the half plane, not to the entire process.

The appeal of causal models is the sequential (autoregressive) coupling of (6.38), allowing very fast estimation using the Kalman filter. However, in practice these models have limited applicability, since most random fields are not well represented causally.

6.5 Gibbs Random Fields

Gibbs random fields (GRFs) are random fields characterized by neighbouring-site interactions. These were originally used in statistical physics [55, 332] to study the thermodynamic properties of interacting particle systems, such as lattice gases, and their use in image processing was popularized by the papers of Besag [25,26] and Geman and Geman [127]. The neighbouring interactions in GRFs lead to effective and intuitive image models, and GRFs are frequently used as prior models in Bayesian formulations.

Mathematically, a Gibbs distribution for a random field $\underline{z}$ is

$$p(\underline{z}) = \frac{1}{\mathbb{Z}} e^{-\beta H(\underline{z})}, \tag{6.39}$$

where $\beta > 0$ is a constant, a "temperature" parameter,[1] and

$$\mathbb{Z} = \sum_{\underline{z}'} e^{-\beta H(\underline{z}')} \tag{6.40}$$

is a normalization constant, known as the *partition function*. The enormity of the sum prevents $\mathbb{Z}$ from being evaluated for all but the tiniest problems. Algorithms which work with GRFs avoid the evaluation of $\mathbb{Z}$ by focusing entirely on likelihood *ratios* or energy *differences*:

$$\frac{p(\underline{z}_1)}{p(\underline{z}_2)} = \frac{e^{-\beta H(\underline{z}_1)}/\mathbb{Z}}{e^{-\beta H(\underline{z}_2)}/\mathbb{Z}} = \exp\Big(-\beta\big(H(\underline{z}_1) - H(\underline{z}_2)\big)\Big). \tag{6.41}$$

The well-known Metropolis and Gibbs Sampler algorithms both use this approach, and are discussed in Section 11.3.

[1] See Simulated Annealing in Section 11.3.1 to understand the role of temperature parameter $\beta = 1/T$ in Gibbs distributions.

Although it would appear that we are focusing once again on probability densities, $p(\underline{z})$ is, in fact, never evaluated. All inferences of $\underline{z}$ take place implicitly, strictly through evaluations of the *energy function* $H(\underline{z})$. The energy function is normally written as a sum of local interaction terms, called *clique potentials*:

$$H(\underline{z}) = \sum_{c \in \mathcal{C}} V\big(\{z_i, i \in c\}\big), \tag{6.42}$$

where $c \subset \Omega$ is a *clique*, either a single site or a set of interacting sites, $V(\cdot)$ is a clique potential, and $\mathcal{C}$ is the set of all possible cliques. Any random field $\underline{z}$ satisfying (6.39),(6.42) is known as a Gibbs random field.

GRFs have been broadly used in statistical image processing, due to their attractive properties:

1. There is a mathematical equivalence between Gibbs and Markov random fields, unifying these two concepts.
2. In the linear-Gaussian case, Gibbs random fields are easily understood in the context of constraint-matrix L, as discussed throughout Chapter 5.
3. A simple, intuitive energy function H can describe enormously complicated probability functions p.
4. Most significantly, there exist algorithms for Gibbs random fields which extend easily to solving nonlinear estimation and sampling problems, problems which become exceptionally complex under classical Bayesian estimation.

   In particular, discrete-state problems, which are inherently nonlinear, are widely solved using Gibbs methods.

We address the above properties in turn. In terms of the relationship between Gibbs and Markov fields, we first need to establish a connection between cliques and neighbourhoods. Suppose that $\mathcal{N}_j$ is a noncausal neighbourhood system on lattice Ω:

$$\mathcal{N}_j \subset \Omega, \qquad j \notin \mathcal{N}_j. \tag{6.43}$$

A neighbourhood system $\mathcal{N}_j$ induces a clique set $\mathcal{C}$, such that any pair of points in $c \in \mathcal{C}$ are neighbours:

$$j, k \in c \quad \Rightarrow \quad j \in \mathcal{N}_k,\ k \in \mathcal{N}_j \tag{6.44}$$

Examples of the clique sets associated with first- and second-order neighbourhoods are shown in Figure 6.10.

Then, by the Hammersley–Clifford theorem [25], the GRF and MRF are equivalent:

> *z is a Markov random field with respect to $\mathcal{N}$ if and only if z is a Gibbs random field with respect to $\mathcal{C}$, where $\mathcal{C}$ is the set of cliques with respect to neighbourhood system $\mathcal{N}$.*

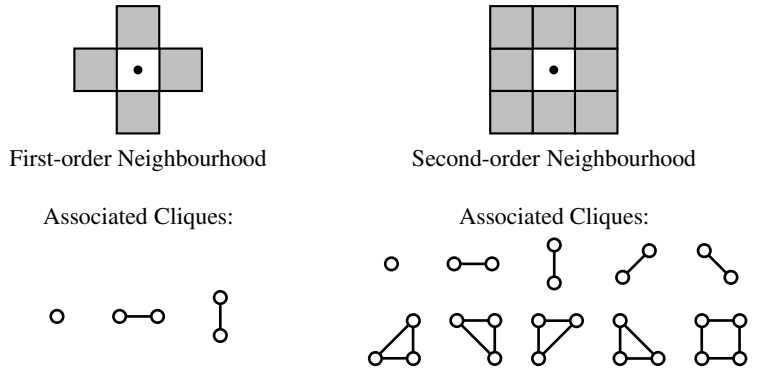


Fig. 6.10. The relationship between Markov neighbourhoods (top) and Gibbs cliques (below). A clique is a set of one or more pixels such that for each pair of pixels, the two pixels are neighbours. Whereas a neighbourhood (shaded) separates a center pixel (dot) from the rest of the domain, cliques have no notion of centre or origin.

In retrospect this should have been obvious. Certainly the connection becomes very clear in the linear-Gaussian case. Suppose we have a random field $\underline{z}$ whose prior is implicitly specified in terms of a set of constraints L:

$$L = \begin{bmatrix} \underline{l}_1^T \\ \vdots \\ \underline{l}_n^T \end{bmatrix} \qquad L\underline{z} = \underline{w} \qquad \underline{w} \sim \mathcal{N}(\underline{0}, I), \tag{6.45}$$

where $\underline{w}$ is white and Gaussian, as in Chapter 5. Then the prior on the random field is

$$p(\underline{z}) = \frac{1}{Z} e^{-\frac{1}{2}\underline{w}^T \underline{w}} \tag{6.46}$$

$$= \frac{1}{Z} e^{-\frac{1}{2}(L\underline{z})^T (L\underline{z})} \qquad = \frac{1}{Z} e^{-\frac{1}{2}\underline{z}^T L^T L \underline{z}} \tag{6.47}$$

$$= \underbrace{\frac{1}{Z} e^{-\frac{1}{2}\sum_i (\underline{l}_i^T \underline{z})^2}}_{\text{Gibbs}} \qquad = \underbrace{\frac{1}{Z} e^{-\frac{1}{2}\underline{z}^T P^{-1} \underline{z}}}_{\text{Markov}} \tag{6.48}$$

That is, a Gauss–Markov random field model directly specifies the inverse-covariance P^{-1}, whereas the Gibbs model implicitly describes the covariance square root through a set of constraints $\{\underline{l}_i\}$ on cliques, where each clique is just the set of pixel locations involved in a given constraint.

Thus the Markov neighbourhood $\mathcal{N}$ is just the union of the convolution of the clique sets $\mathcal{C}$, in precisely the same way that the kernel $\mathcal{Q}$ was found as the convolution of

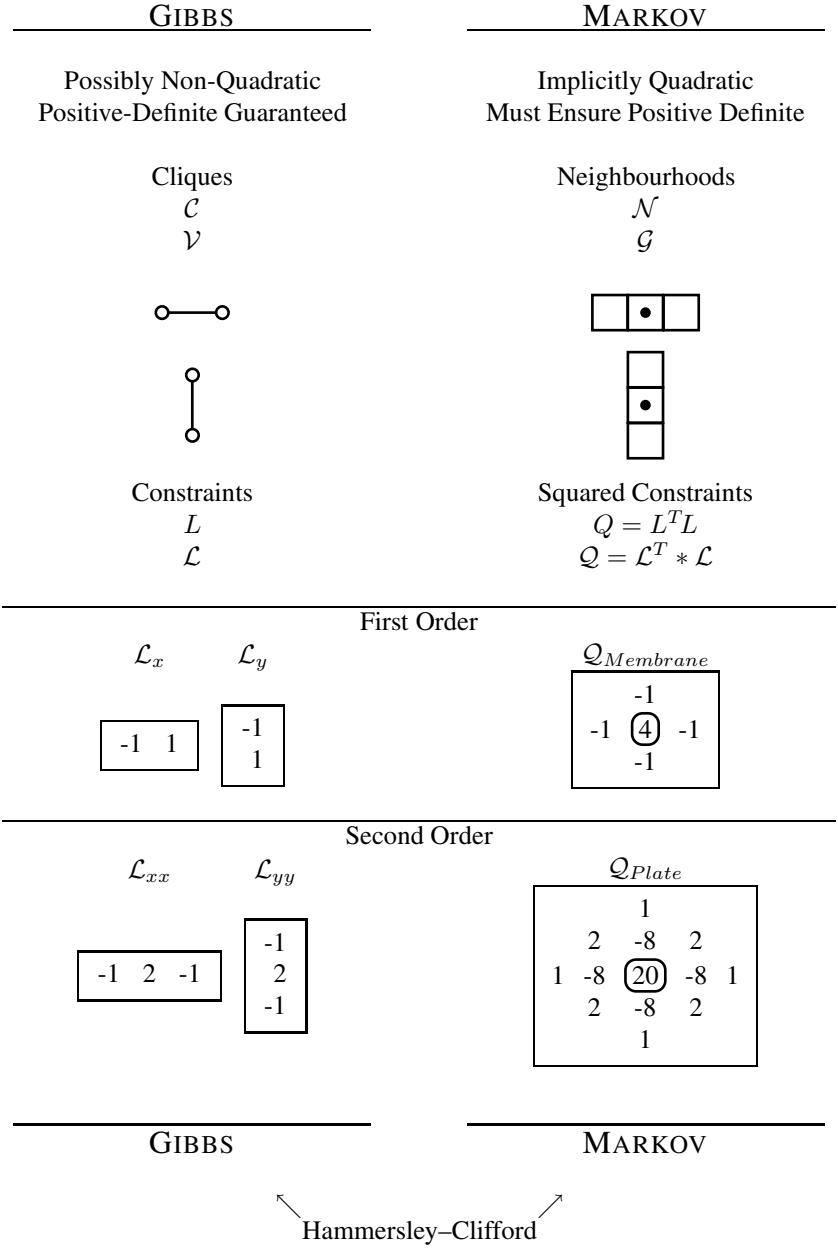


Fig. 6.11. The Hammersley–Clifford theorem establishes a connection between the clique-Gibbs-constraint and the neighbourhood-Markov-squared-constraint contexts.

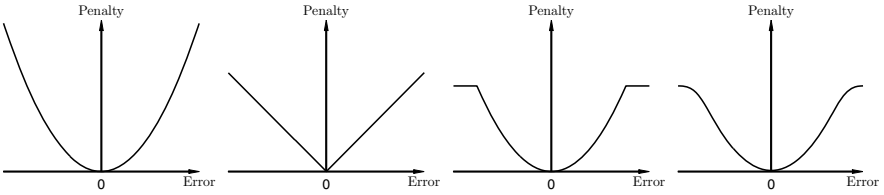


Fig. 6.12. Four possible criteria, relating the degree of penalty to the size of error. The standard quadratic criterion, left, grows rapidly, appropriate for Gaussian statistics (where large deviations from the mean are rare), but inappropriate for problems involving outliers. An illustration of robust estimation is shown in Example 6.2.

constraints $\mathcal{L}$ in Figure 5.8 on page 152. Indeed, the forms of the first-order membrane constraints and kernel in Figure 5.8 are clearly evident in the corresponding cliques and neighbourhood in Figure 6.10. The connections between these concepts are illustrated in Figure 6.11.

So what sorts of local energy functions H are available to implicitly determine the random-field distribution $p(\underline{z})$? Certainly all of the basic smoothness models of Chapter 5, such as membrane, thin-plate, and other combinations of nth-order penalties, are all Gibbs. For example,

$$H(z) = \sum_{i,j} (z_{i,j} - z_{i-1,j})^2 + (z_{i,j} - z_{i,j-1})^2 \tag{6.49}$$

is a membrane energy function, implicitly describing a membrane prior for $p(z)$. Just as simply, introducing measurements to (6.49) leads to

$$H(z|m) = \sum_{i,j} (z_{i,j} - m_{i,j})^2 + (z_{i,j} - z_{i-1,j})^2 + (z_{i,j} - z_{i,j-1})^2 \tag{6.50}$$

which now implicitly describes the *posterior* $p(z|m)$.

However, the real strength of the Gibbs approach lies not in the representational power of H, which is not so different from what we have seen before, rather that there exist methods of sampling and estimation which apply to non-quadratic H. In particular, there are two broad energy classes of great interest:

NON-QUADRATIC RANDOM FIELDS:
Quadratic, or least-squares, criteria are used throughout most of this text because of the resulting simplicity: a quadratic criterion has a unique minimum which can be found linearly.

However in the Gibbs context it is easy to introduce a non-quadratic criterion $J(a, b)$. For example, to include a non-quadratic penalty term on measurement residuals, we could rewrite (6.49) as

$$H(z) = \sum_{i,j} J(z_{i,j}, m_{i,j}) + \sum_{i,j} (z_{i,j} - z_{i-1,j})^2 + (z_{i,j} - z_{i,j-1})^2 \qquad (6.51)$$

where three typical non-quadratic penalty terms are shown in Figure 6.12, and the use and effect of such a term is illustrated in Example 6.2.

Of course, *strongly Markov* random fields can also possess nonlinear features, in principle, however the strength of Gibbs random fields is that relatively efficient methods of sampling and estimation exist for such nonlinear models, as shown in Chapter 11.

DISCRETE-STATE RANDOM FIELDS:

The discrete-state case reflects the origins of Gibbs fields in quantum physics where each field element reflects a discrete quantum state, such as the spin of an atom. The most famous of all such models is the binary Ising model [335]

$$H(z) = \sum_{i,j} z_{i,j} z_{i,j-1} + \sum_{i,j} z_{i,j} z_{i-1,j} \qquad z_{i,j} \in \{-1, +1\}. \qquad (6.52)$$

The simple generalization of the binary Ising form to a q-state element is known as the Potts model:

$$H(z) = \sum_{i,j} \delta_{z_{i,j}, z_{i,j-1}} + \sum_{i,j} \delta_{z_{i,j} z_{i-1,j}} \qquad z_{i,j} \in \{0, 1, \ldots, q-1\}. \qquad (6.53)$$

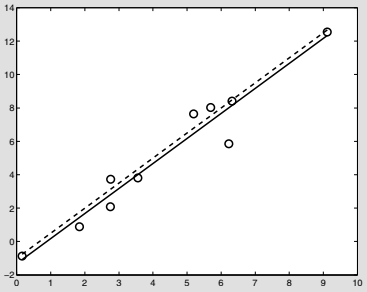
In the context of image modelling, discrete states are most often used in hidden models, where the behaviour of each pixel in an image is governed by one of a finite set of possible states, such as in a mixture model. Two common examples are image segmentation (Appendix C), the labelling of each image pixel as belonging to one of a finite set of segments, or image classification, associating each pixel with one of a set of possible classes. Hidden state models are discussed further in Chapter 7.

The choice between Gibbs or Markov random fields normally proceeds fairly obviously from the given problem. If we seek to learn a linear, second-order model from given data, then the Markov formulation is straightforward. If we wish to "guess" a model or develop it heuristically, or if our model contains nonlinearities or discrete random variables, then the Gibbs formulation is to be preferred.

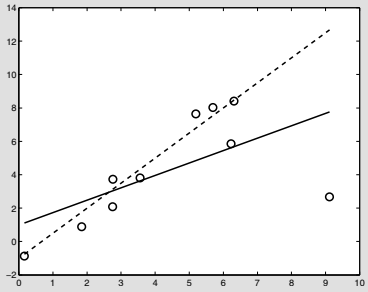
Example 6.2: Robust Estimation

We present here a very simple example, illustrating the use of non-quadratic criteria in estimation. Suppose we have a simple non-Bayesian parameter estimation problem, a linear regression problem given ten data points (see Example 3.1).

If the measurement error is Gaussian with constant variance (left), then the least-squares criterion is appropriate and gives an accurate result. If the measurements have the occasional bad data point (right), inconsistent with a Gaussian distribution, then the regression result (solid) is poor in comparison with truth (dashed).



Regression with Gaussian Noise

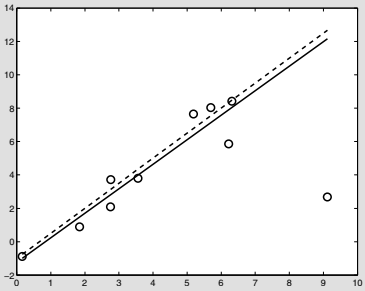


Regression with Non-Gaussian Noise

Instead, to limit the effect of the bad data point, we can compare a measurement m and its predicted location $\hat{m} = C\underline{\hat{z}}$ using a non-quadratic criterion, such as the capped penalty from Figure 6.12

$$J(m, \hat{m}) = \min\{\zeta^2, (m - \hat{m})^2\}$$

for some threshold ζ. The resulting estimates, below, are generated with a threshold $\zeta = 4$,



and are clearly superior to the previous results. This estimator is nonlinear, and although easily implemented for linear regression, would generally be difficult for large spatial problems. That the Gibbs sampler can handle such non-quadratic criteria is a strong asset.

6.6 Model Determination

In many problems we assert a constraints matrix L, such as the membrane and thin-plate priors from Chapter 5. In most cases the constraints asserted in L are local, involving the interactions between nearby state elements, in which case the resulting $L^T L$ is a sparse banded matrix and we have implicitly specified a Markov prior for the random field. Indeed, in any problem in which we have local constraints, or in which the covariance inverse is modelled as sparse-banded, we should immediately understand the implied prior to be Markov.

However, we are not necessarily given a banded inverse matrix or a set of constraints; rather in many cases we are presented with a sample random field, from which we would like to infer a Markov model. We can, of course, compute a sample covariance from the given random field, however we run into two difficulties:

1. The matrix inverse of the sample covariance will normally not be a nice, banded matrix. In most cases, the given random field is not *exactly* Markov, but we wish to approximate it as such. Moreover, even if the given random field is precisely Markov, the *sample* statistics may not be.
2. Any sample covariance is guaranteed to be positive-semidefinite. If the sample covariance is furthermore positive-definite then we can compute its matrix inverse, which will also be positive-definite, however truncating the matrix inverse to make it banded may *not* leave it positive-definite (see Problem 5.3).

So our task is one of model approximation. Given sample statistics P, find a matrix G such that

1. Markovianity: G is sparse-banded
2. Validity: G is positive-definite
3. Fit to Model: $G^{-1} \approx P$.

A Markov random field is essentially a noncausal, multidimensional version of an autoregressive process, thus we begin by discussing model learning for autoregressive processes, before tackling the related, but more complicated, problem of learning for Markov processes.

6.6.1 Autoregressive Model Learning

Autoregressive models are a classic, long-established form of time-series modelling [6, 37], in which we wish to approximate a given stationary time series $z(t)$ in terms of an nth-order autoregressive process

$$z(t) = \sum_{i=1}^{n} \alpha_i z(t-i) + w(t), \tag{6.54}$$

where the zero mean, white noise process w is uncorrelated with the past of z:

$$E\big[w(t)\big] = 0 \quad E\big[w(t)w(s)\big] = \sigma^2 \delta_{s,t} \quad E\big[w(t)z(s)\big] = 0 \text{ if } s < t. \tag{6.55}$$

The process z in (6.54) is assumed to be zero-mean; generalization to the nonzero-mean case is straightforward.

Because the coefficients α_i affect the relationship between $z(t)$ and $z(t-i)$, it seems plausible to suppose that the coefficients can be estimated by examining the correlation of z with shifted versions of itself. Since z is stationary, its statistics can be captured by a correlation kernel

$$\mathcal{P}_j = E\big[z(t)z(t-j)\big] \equiv E\big[z(t-j)z(t)\big] = \mathcal{P}_{-j} \tag{6.56}$$

so by substituting (6.54) into (6.56) we get

$$\mathcal{P}_j = E\big[z(t)z(t-j)\big] = E\left[\left(\sum_{i=1}^{n} \alpha_i z(t-i) + w(t)\right) z(t-j)\right] \tag{6.57}$$

$$= \sum_{i=1}^{n} \alpha_i E\big[z(t-i)z(t-j)\big] + E\big[w(t)z(t-j)\big] \tag{6.58}$$

$$= \sum_{i=1}^{n} \alpha_i \mathcal{P}_{i-j} + E\big[w(t)z(t-j)\big], \tag{6.59}$$

where the correlation $E\big[w(t)z(t-j)\big]$ between z and w follows from (6.55):

$$\begin{aligned} j > 0 &\Longrightarrow E\big[w(t)z(t-j)\big] = 0 \\ j = 0 &\Longrightarrow E\big[w(t)z(t-j)\big]\Big|_{j=0} = E\big[w(t)\big(\textstyle\sum_{i=1}^{n} \alpha_i z(t-i) + w(t)\big)\big] \\ &\qquad\qquad = 0 + E\big[w(t)w(t)\big] = \sigma^2. \end{aligned} \tag{6.60}$$

Therefore we arrive at a set of linear equations, known as the Yule–Walker equations [37, 252], which interrelate the lag-correlations and the autoregressive coefficients:

$$\mathcal{P}_j = \sum_{i=1}^{n} \alpha_i \mathcal{P}_{i-j} + \delta_j \sigma^2. \tag{6.61}$$

For this system of equations to be invertible we need $n+1$ lag correlations, to infer the n AR coefficients plus the noise variance σ^2.

We can vectorize (6.61) by stacking the correlations and the AR coefficients for $i, j = 1, \ldots, n$:

$$\begin{bmatrix} \mathcal{P}_1 \\ \vdots \\ \mathcal{P}_n \end{bmatrix} \equiv \underline{p} = \mathfrak{P}\underline{\alpha} \equiv \begin{bmatrix} \mathcal{P}_0 & \mathcal{P}_{-1} & \cdots & \mathcal{P}_{-(n-1)} \\ \mathcal{P}_1 & \mathcal{P}_0 & \cdots & \mathcal{P}_{-(n-2)} \\ \vdots & \vdots & & \vdots \\ \mathcal{P}_{n-1} & \mathcal{P}_{n-2} & \cdots & \mathcal{P}_0 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix} \tag{6.62}$$

Given a sample process $\check{z}(t)$ we can find estimates of the lag correlations

$$\hat{\mathcal{P}}_j = \frac{1}{N} \sum_{t=1}^{N} \check{z}(t)\check{z}(t-j) \tag{6.63}$$

from which we can solve for the AR coefficients $\underline{\alpha}$ in (6.62) as

$$\underline{p} = \mathfrak{P}\underline{\alpha} \quad \Longrightarrow \quad \underline{\hat{\alpha}} = \hat{\mathfrak{P}}^{-1}\underline{\hat{p}} \tag{6.64}$$

and where the noise variance is estimated from autocorrelation lag $j = 0$:

$$\hat{\sigma}^2 = \hat{\mathcal{P}}_0 - \underline{\hat{\alpha}}^T \underline{\hat{p}}. \tag{6.65}$$

The choice of n, the model order, cannot be estimated and must be asserted. In the same way that a higher-order polynomial can always improve the goodness of fit to a given number of points, similarly the variance of the autoregressive residuals

$$\text{var}\left(z(t) - \sum_{i=1}^{n} \alpha_i z(t-i) \right) \tag{6.66}$$

is monotonically decreasing as n increases, therefore other criteria must be used to limit n.

6.6.2 Noncausal Markov Model Learning

The derivation of the Markov model is very similar to that, above, for the autoregression coefficients. The reasons for this similarity should be made clear by an examination of Table 6.1.

Our goal is as in the autoregressive case: we wish to find a stationary Markov model of specified neighbourhood which fits the given data [59–61, 199]. The noncausal Markov random field is specified by coefficients $\bar{g}$ as

$$z_j = \sum_{i \in \mathcal{N}_j} \bar{g}_{i-j} z_i + w_j, \tag{6.67}$$

where the driving noise w is zero mean, *not* white, and uncorrelated with the entire random process $z_i, i \neq j$.

	Autoregressive Model	Noncausal MRF Model
Model	$z(t) = \sum_{i=1}^{n} \alpha_i z(t-i) + w(t)$ (6.54)	$z_j = \sum_{i \in \mathcal{N}_j} \bar{g}_{i-j} z_i + w_j$ (6.19)
Model to Noise	$E\big[w(t)z(s)\big] = 0$ if $s < t$ (6.55)	$E[z_i w_j] = 0 \ \ \forall j \neq i$ (6.19)
Noise	$E\big[w(t)w(s)\big] = \delta_{s,t}\sigma^2$ (6.55)	$E[w_j w_i] = \bar{g}_{i-j}\sigma^2$ (6.24)

Table 6.1. A comparison of the formulation and assumptions for stationary autoregressive models, left, and stationary Markov random fields, right. The two models are similar, however there are subtle differences in the noise assumptions, in that the MRF noise is correlated, not white, and uncorrelated with the whole of the random field, rather than the one-sided uncorrelation in the autoregressive case.

Because the Markov coefficients $\bar{g}_{i-j}$ affect the relationship between z_i and z_j, it again seems plausible to suppose that the coefficients can be estimated by examining the correlation of z with shifted versions of itself. Since z is stationary, its statistics can be captured by a correlation kernel

$$\mathcal{P}_{k-j} = E\big[z_j z_k\big] \equiv E\big[z_k z_j\big] = \mathcal{P}_{j-k} \tag{6.68}$$

so by substituting (6.67) into (6.68) we get

$$\mathcal{P}_j = E\left[z_0 z_j\right] = E\left[\left(\sum_{i \in \mathcal{N}_0} \bar{g}_i z_i + w_0\right) z_j\right] \tag{6.69}$$

$$= \sum_{i \in \mathcal{N}_0} \bar{g}_i E\big[z_i z_j\big] + E\big[w_0 z_j\big] \tag{6.70}$$

$$= \sum_{i \in \mathcal{N}_0} \bar{g}_i \mathcal{P}_{j-i} + \delta_{0,j}\sigma^2. \tag{6.71}$$

Once again we arrive at a set of linear equations, the analogue of the Yule–Walker equations for the noncausal multidimensional case:

$$\mathcal{P}_j = \sum_{i \in \mathcal{N}_0} \bar{g}_i \mathcal{P}_{j-i} + \delta_{0,j}\sigma^2. \tag{6.72}$$

Let $n = |\mathcal{N}_0|$ be the number of neighbours, and $(\mathcal{N}_0)_i$ the relative index of the ith neighbour; then (6.72) can be vectorized, as before:

$$\begin{bmatrix} \mathcal{P}_{(\mathcal{N}_0)_1} \\ \vdots \\ \mathcal{P}_{(\mathcal{N}_0)_n} \end{bmatrix} \equiv \underline{p} = \mathfrak{P}\underline{\bar{g}} \equiv \begin{bmatrix} \mathcal{P}_{(\mathcal{N}_0)_1-(\mathcal{N}_0)_1} & \cdots & \mathcal{P}_{(\mathcal{N}_0)_1-(\mathcal{N}_0)_n} \\ \vdots & & \vdots \\ \mathcal{P}_{(\mathcal{N}_0)_n-(\mathcal{N}_0)_1} & \cdots & \mathcal{P}_{(\mathcal{N}_0)_n-(\mathcal{N}_0)_n} \end{bmatrix} \begin{bmatrix} \bar{g}_{(\mathcal{N}_0)_1} \\ \vdots \\ \bar{g}_{(\mathcal{N}_0)_n} \end{bmatrix} \tag{6.73}$$

Given sample data $\underline{\check{z}}$ we can compute estimates of the offset correlations

$$\hat{\mathcal{P}}_j = \frac{1}{N}\sum_{i=1}^{N} \check{z}_i \check{z}_{i+j} \tag{6.74}$$

from which we can solve for the MRF coefficients from (6.73) as

$$\underline{p} = \mathfrak{P}\underline{\bar{g}} \quad \Longrightarrow \quad \underline{\hat{\bar{g}}} = \hat{\mathfrak{P}}^{-1}\underline{\hat{p}} \tag{6.75}$$

and where the noise variance is estimated from autocorrelation offset $j = 0$:

$$\hat{\sigma}^2 = \hat{\mathcal{P}}_0 - \underline{\hat{\bar{g}}}^T \underline{\hat{p}}. \tag{6.76}$$

As in the autoregressive case, where the model order n could not be inferred, similarly for Markov random fields it is not possible to infer the neighbourhood $\mathcal{N}$. The choice of $\mathcal{N}$ must either be asserted, or we can try to look at the model error

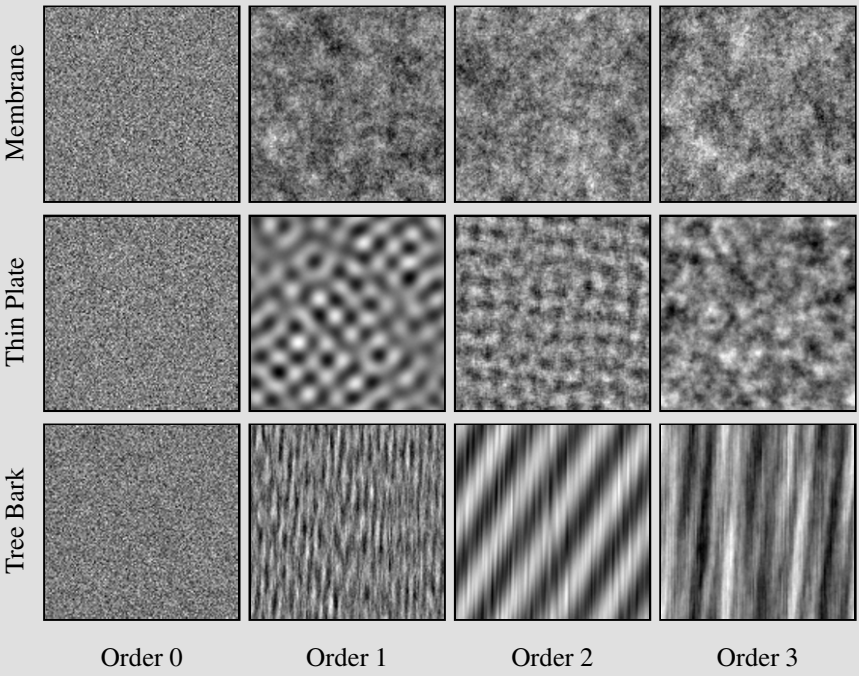
$$\text{var}\left(z_j - \sum_{i \in \mathcal{N}_j} \bar{g}_{i-j} z_i \right) \tag{6.77}$$

as a function of neighbourhood size, and choose the smallest neighbourhood which gives an acceptable level of error. Example 6.3 illustrates the process of Markov model estimation.

Example 6.3: Markov Model Inference and Model Order

The images below illustrate the learning of MRF models for three different prior models.

Given each of the three textures from Example 6.1, a MRF prior model is learned for each of four different neighbourhood orders. The following image panels show the sample texture synthesized from each of the learned models:



By definition, the zero-order models have no spatial relations, and therefore are all white noise.

The true membrane prior is first order, and is therefore learned adequately well with a first-order model, with no substantial changes at higher orders.

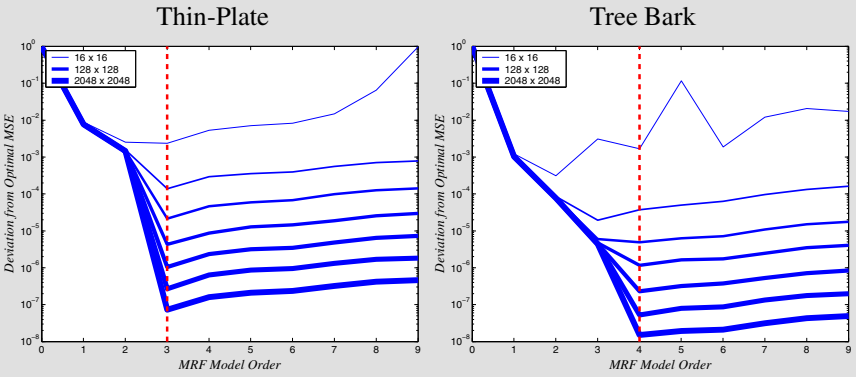
The true thin-plate prior is third order, and the inadequacy of the first- and second-order reconstructions can clearly be seen.

Finally the true "Tree-Bark" model is fourth order and poorly conditioned, and so is not properly learned above, although the third-order model is able to represent the vertical banding.

Example continues . . .

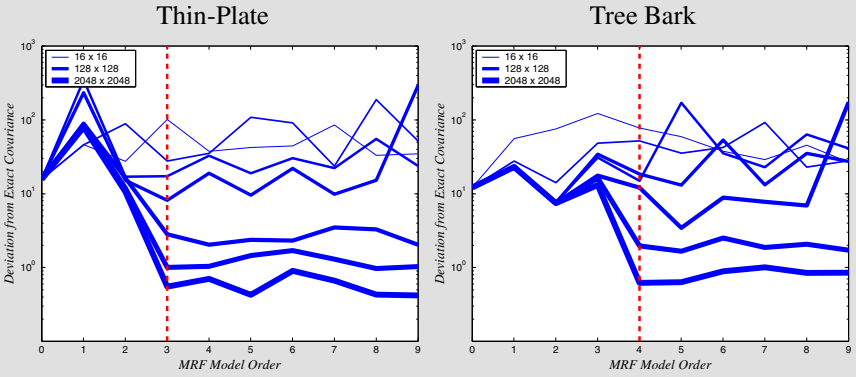
Example 6.3: Markov Model Inference and Model Order (cont'd)

The following two figures plot the model error variance (6.77) as a function of model order and the size of the sample image from which to learn the model:



We can see how misleading the model error variance (6.77) can be as an indicator of model accuracy. Going from a zeroth- to first-order model, the thin-plate MSE dropped from 1.0 to 0.01, and the "Tree-Bark" MSE from 1.0 to 0.001, however in both cases the first-order reconstructions are quite poor.

Instead, we could consider looking at the covariance of the learned model, and compare that to the covariance of the underlying true model. The following figures plot the difference in the learned and underlying covariance kernels, based on a weighted sum of absolute kernel differences:

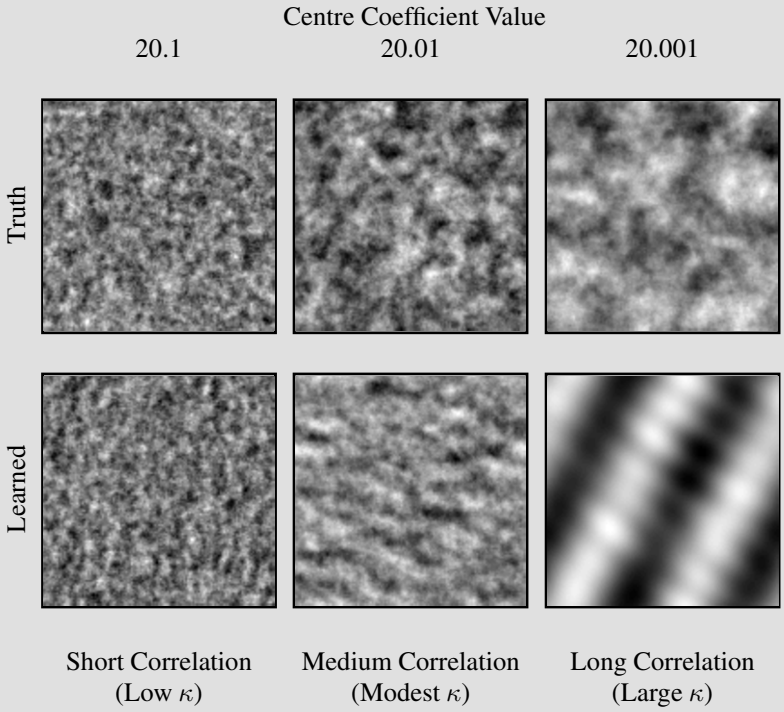


We can now clearly see the failure of the first-order models to learn the correct field statistics, most strikingly in the thin-plate case, and also how a relatively large sample image is required before we can reliably say to have learned the true model.

Example 6.4: Model Inference and Conditioning

We saw in Example 6.3 how the assumed model order affected the quality of the inferred result. However, presumably the ability to learn a model might also depend on the conditioning of the true model, in that a prior with high condition number κ requires the Markov model G to be learned *just right*, a hair's-breadth away from singularity, whereas a model with low condition number would be more tolerant of parameter variations.

Consider the following, in which three thin-plate models are learned having different correlation lengths (and condition number) based on the value of their center element. All of the sample images are 128×128 in size:

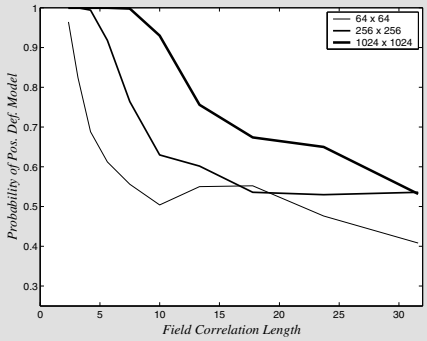


We can clearly see the reduction in performance as we move from well conditioned (short correlation length, left) to poorly conditioned (right).

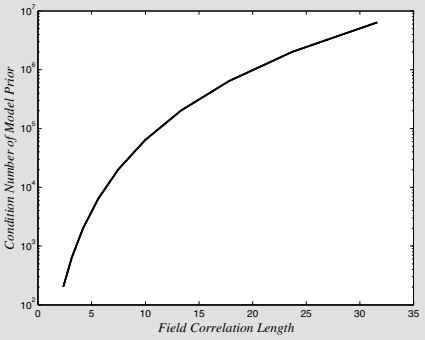
The plots on the facing page summarize this, plotting the probability of finding a positive-definite model as a function of prior model correlation length and the given size of the sample image:

Example continues . . .

Example 6.4: Model Inference and Conditioning (cont'd)



Probability that the Learned Model is Positive-Definite



Condition Number of True Model

It is important to recognize that a set of model parameters g which fails to be positive-definite is *not* inherently useless. These model parameters could still be used as a feature for texture discrimination in pattern recognition, or texture segmentation in image processing, for example. However, as a *statistical prior* such a model would have no validity.

6.7 Choices of Representation

The parallels between Gibbs and Gauss–Markov fields, and the related parallels with the models introduced in Chapter 5, motivate a renewed examination of the question of representation, following the discussion of Section 5.8.

As is illustrated in Figure 6.13, we have a total of twelve options, laid out along three independent axes:

1. Specifying constraints directly or in squared form;
2. Specifying a convolutional kernel versus a full form;
3. Selecting a deterministic model versus a regular or inverse statistical one.

The comparative tradeoffs and issues are mostly the same as in Section 5.8, and do not need repeating here. What is new is the set of Gibbs–Markov models $(\mathcal{G}, G, \mathcal{V}, V)$, which neatly straddle deterministic models $(\mathcal{Q}, Q, \mathcal{L}, L)$ and the statistical dynamic models $(\mathcal{P}, P, \mathcal{A}, A)$:

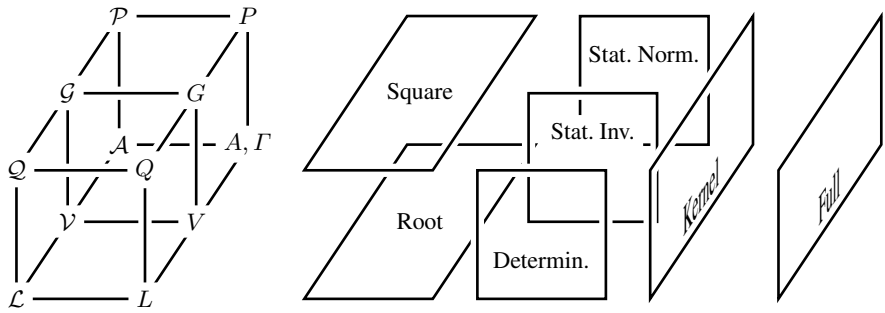


Fig. 6.13. Building on Figure 5.19, we now have twelve forms of representation, left, where the forms differ based on the choice of three underlying aspects: representing a model or its square root, full or convolutional-kernel, and deterministic or statistical modelling. The middle set of inverse models $(\mathcal{G}, \mathcal{V}, G, V)$ are those contributed by random field modelling.

1. Interpreted as inverse-covariances, the Gibbs–Markov models have a straightforward interpretation as representing the inverse-statistics of a covariance or dynamic model.

2. Viewed as equivalent, by analogy, with the deterministic constraint models, the Gibbs–Markov models give a clear, statistical interpretation to local/sparse models which were previously much less explicit.

3. The Gibbs model extends the previous dynamic $(\mathcal{A}, A)$ and constraint $(\mathcal{L}, L)$ formulations by allowing more general classes of models, including discrete-valued states and non-quadratic error criteria.

Application 6: Texture Classification

The problem of texture classification is exceptionally well studied, and enjoys a literature spanning more than two decades [60, 225, 226, 241, 242, 258–260].

Suppose that we have a number of textures, for example the five samples from the widely-studied Brodatz set [45] shown in Figure 6.14.

We assume these textures to be known, making this a *supervised* classification problem [91]. We can learn a Markov random field model for each of these textures, as was done in Example 6.3. Then, given an image to segment into its constituent textures, we can attempt to relate the unknown image to the learned models.

There are many ways in which such a classification can be undertaken, however let's consider a very basic approach. From each texture image I_t we extract a number

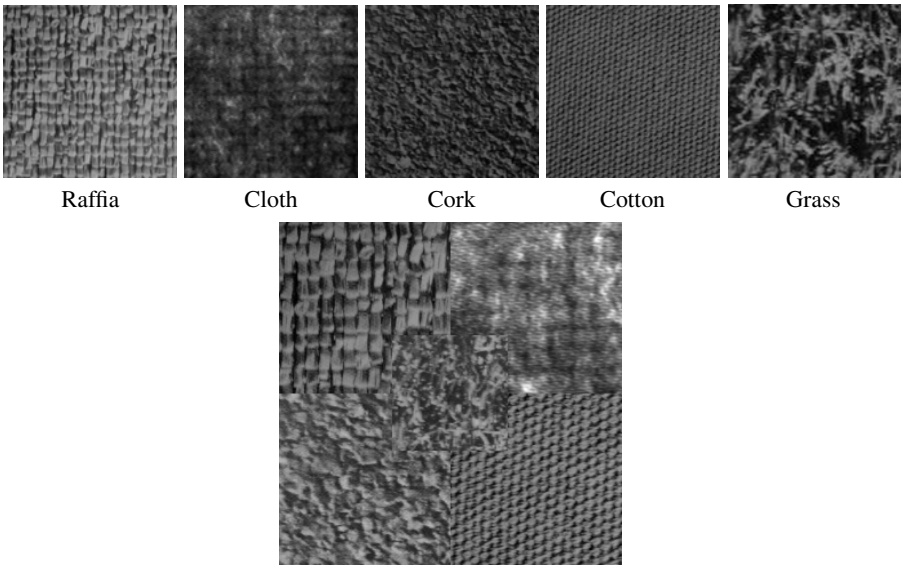


Fig. 6.14. Given five sample Brodatz textures [45], top, we wish to use a random field model to segment a given composite image, below.

of samples $\{S_{t,i}\}$ of size $N \times N$, and a Markov model $g(S)$ of order q is learned from each sample. Then, given an unknown sample $\bar{S}$, we learn its model $g(\bar{S})$, and classify the texture

$$\hat{t}(\bar{S}) = \arg_t \min \left\| g(\bar{S}), \left\{ g(S_{t,i}) \right\} \right\| \tag{6.78}$$

where the norm $\| \cdot \|$ needs to assess the closeness of the unknown model g to the learned sets. The simplest norm is a Euclidean nearest-neighbour [91] approach, treating the Markov model as a vector of GMRF coefficients:

$$\hat{t}(\bar{S}) = \arg_t \min \left\{ \min_i \left| g(\bar{S}), \left\{ g(S_{t,i}) \right\} \right|^2 \right\}. \tag{6.79}$$

Figure 6.15 plots the probability of a correct texture match, as a function of the texture patch size N, and as a function of the size of the Markov model neighbourhood. Clearly the probability of a correct match increases with N, since a larger sample more distinctly characterizes a given texture, however the *lower* performance with higher model order may be surprising: although a higher model order *does* represent a given texture better, the increased number of parameters in the model leads to much greater variability in g and a poorer classification.

Selecting a patch size of 30×30 and a second-order (3×3 neighbourhood) Markov model, with the nearest-neighbour classifier of (6.79), leads to the results as shown in Figure 6.16(a). Because each texture patch is processed independently of the oth-

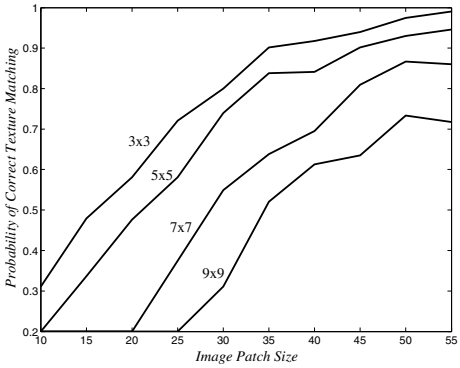
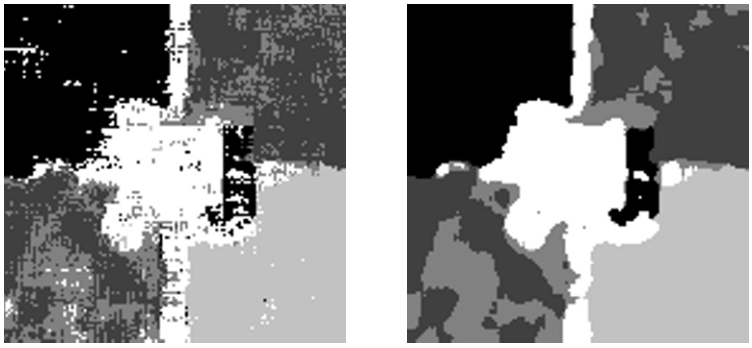


Fig. 6.15. The probability of matching a texture sample of size $N \times N$ to its correct class, for four Markov neighbourhood sizes. More local models and larger texture patches lead to improved matching probability.



(a) Segmented Results (b) Median Filtered

Fig. 6.16. The segmented composite texture of Figure 6.14. When each texture patch is segmented individually, left, there are no spatial constraints and the resulting classification is somewhat noisy. A median filter can be used to assert smoothness in the classification, right.

ers, there is no spatial constraint and the estimates appear noisy. Essentially what we would like here is a prior model on the underlying texture label; such an underlying prior model is a *hidden* model, the subject of Chapter 7. A very simple spatial constraint is asserted by a median filter [54], which generates the imperfect, but credible, results shown in Figure 6.16(b).

Summary

Markovianity means a conditional separation, the separation of two parts by some boundary:

1D : For one-dimensional problems, one or more values of the random process form the boundary separating past from future;

dD : For multidimensional problems, it is easier to talk about the boundary required to separate a single pixel (inside) from the entire rest of the domain (outside).

In separating a single pixel from the rest of the domain, the boundary which affects this separation is the *neighbourhood* $\mathcal{N}$. The size of the neighbourhood is the model *order*. There are two classes of models:

1. Causal / acyclic models, in which the state elements can be ordered, leading to efficient algorithms but relatively poor models.
2. Noncausal / loopy models, in which there is no ordering, leading to greater complexity, but also better modelling. Nearly all of the examples in the text are drawn from the noncausal case.

If a Markov field obeys Gaussian statistics, we have the important class of *Gauss–Markov Random Fields* (GMRFs),

$$z_j = \sum_{k \in \mathcal{N}_j} \bar{g}_{j,k} z_k + w_j \qquad E[z_k w_j] = 0 \;\; \forall j \neq k, \tag{6.80}$$

characterized by model parameters $\bar{g}$. If the random field is stacked into a vector, and the model parameters similarly into a matrix, then (6.80) is neatly rewritten as

$$G\underline{z} = \underline{w} \qquad E\left[\underline{z}\,\underline{w}^T\right] \text{ diagonal.} \tag{6.81}$$

The correlation of the noise structure is determined by G,

$$\text{cov}(\underline{w}) \Longleftrightarrow G \tag{6.82}$$

and the model G is interpreted as an *inverse* covariance:

$$\text{cov}(\underline{z}) \Longleftrightarrow G^{-1}, \tag{6.83}$$

so that assuming an inverse covariance to be sparse and banded is equivalent to assuming that the random field is Markov.

The four essential facts of GMRFs:

1. The GMRF model parameters are essentially the matrix-inverse of the covariance. Therefore the matrix-inverse of a Markov covariance will be highly sparse.

2. The driving noise process $\underline{w}$ is *not* white. If we assert that the noise is uncorrelated with the random field,
$$E\left[\underline{z}\,\underline{w}^T\right] = \sigma^2 I \tag{6.84}$$
then it is *not* possible to also assert that the noise is white.

3. The *scale* of the random field is controlled by the model parameters $\bar{g}$.

4. The *complexity* of the random field is controlled by the model *order*.

For Further Study

The question of what happens to Markov random fields when cast into a hierarchical context is looked at in Section 8.5.

Markov fields are often used in pairs, with a hidden or underlying field serving to condition a second, visible field. The use of and modelling with hidden fields will be discussed in Chapter 7.

The fastest and easiest ways to get started with random fields is by using fast Fourier transforms (FFTs). The connection between Markov random fields and FFTs is explored in Section 8.3. Because Markov random fields imply a sparse P^{-1} they are compatible with iterative approaches to estimation, which are developed in Chapter 9.

The texts by Winkler [335], Li and Gray [205], and Won and Gray [336] all comprehensively discuss random fields, with Won and Gray focusing on Markov fields, and Winkler more on Gibbs fields.

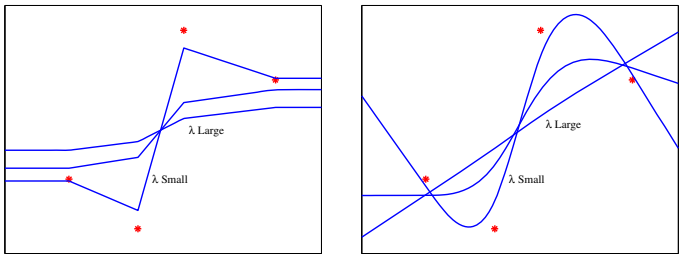
Sample Problems

Problem 6.1: Markov Order

(a) For each of the panels in Figure 6.4, what is the maximum order of the corresponding Markov field if the field can be separated by the given boundary?
(b) Construct three 2D boundaries, like the ones drawn in Figure 6.4, such that the maximum order of the corresponding Markov field is second-, third-, and fourth-order.

Problem 6.2: Membrane, Thin-Plate, and Markovianity

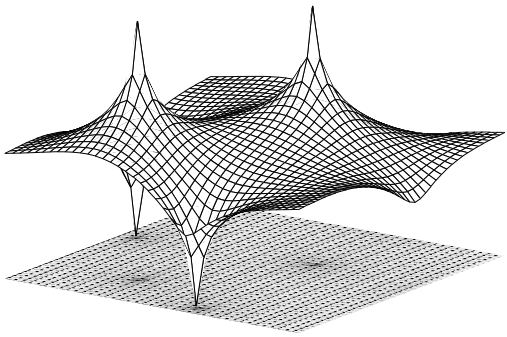
Below are two figures, reproduced from Example 2.7, back on page 36:



where we generated estimation results for one-dimensional processes given first- or second-order constraints.

(a) Why is the first-order constraint Markov? What order of Markovianity does it possess?
(b) Why is the second-order constraint Markov? What order of Markovianity does it possess?
(c) The first-order results look piecewise-linear, leading to two questions:
- Explain convincingly why the piecewise-linearity is due to first-order Markovianity.
- Does this mean that the estimate at a point depends only on the two closest measurements, or not?

(d) The figure below shows a two-dimensional first-order estimate, based on the measurements and plotting arrangement as in Example 5.1:



Clearly these estimates are *not* piecewise-linear, although the prior is still first-order Markov. What is it about the difference between one- and two-dimensional problems that leads to the difference in behaviour?

Problem 6.3: Markov Model Inference and Conditioning

We know two things:

1. The worse the conditioning of a problem, the more sensitive the resulting estimates can be to model errors.

2. The smaller a given sample random field, the greater the errors in the inferred MRF model.

Therefore we might expect there to be an interrelationship between kernel conditioning and the amount of data needed to learn the kernel model.

Consider the thin-plate kernel in Example 6.1, where we set the central element of the kernel to one of four possible values: 20.1, 20.03, 20.01, 20.003. The conditioning of the kernel becomes worse as this central element is decreased (the kernel becomes singular, corresponding to a condition number of ∞, when the central element is set to 20).

You will need to use the FFT method of Section 8.3, or access the sample MATLAB code available online, in order to synthesize the sample random fields.

For each of the four kernels:

(a) Generate a random $N \times N$ sample, using the FFT, for some choice of N, using the true kernel.
(b) Learn a fourth-order MRF model from the $N \times N$ sample.
(c) Now generate a random 128×128 sample, using the FFT with the kernel just learned.
(d) Repeat the above three steps, experimenting with differences choices of N, seeing what N is required to generate reasonable results in (c).

Comment on your observations.

Problem 6.4: Open-Ended Real-Data Problem — Texture Classification

There is an extensive literature [60, 62, 226, 260] on the use of Markov random fields in texture classification. Select an MRF-based method and apply it to a standard texture database, such as the Brodatz, CUReT, UIUC, or KTH-TIPS databases, all of which can be found on the Internet.

Compare the classification accuracy which you obtain using a Markov approach with the published Markov methods [60, 62, 226, 260], and then also with more recently-published approaches, for example those based on local binary patterns [242] and local image patches [315].

7

Hidden Markov Models

Working with nonstationary and heterogeneous random fields presents an interesting challenge. In principle, the modelling methods of Chapter 5 do allow us to construct a nonstationary model, under the assumption that the model boundaries are known.

For example, given a noisy image M that we wish to segment into its piecewise-constant parts, we could attempt to learn a nonstationary model L_{NS} from edge-detection on M, and then estimate $\hat{Z}$ based on model L_{NS}. It is then possible that the pattern of nonstationarity will be more evident in $\hat{Z}$ rather than M, leading to an alternating iterative method:

$$M \overset{\text{Infer}}{\Longrightarrow} L_{NS}^{(1)} \overset{\text{Estimate}}{\Longrightarrow} \hat{Z}_1 \overset{\text{Infer}}{\Longrightarrow} L_{NS}^{(2)} \overset{\text{Estimate}}{\Longrightarrow} \hat{Z}_2 \cdots \tag{7.1}$$

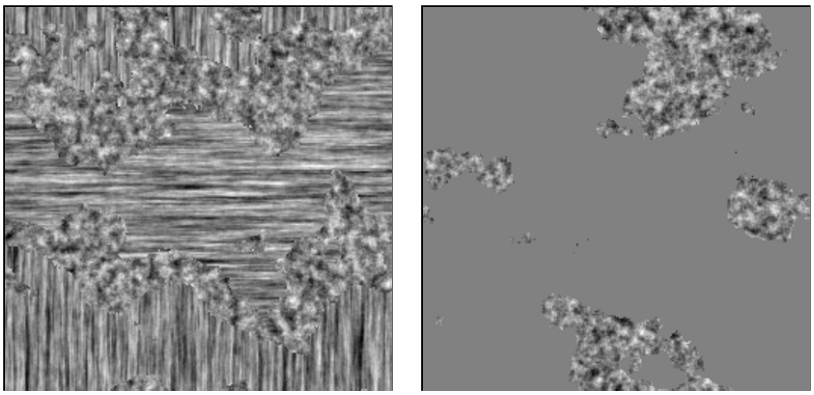
The approach of (7.1) is actually reasonably close to the method we eventually propose for such a problem. However, rather than a heuristic guess at L_{NS}, we should prefer a more systematic formulation.

Consider the two panels in Figure 7.1. The left panel shows a multi-texture image, such that each of the individual textures (thin-plate, wood-grain) are Markov,[1] however the resulting image is not, since a pixel at the boundary between two textures needs more than just a local neighbourhood to understand its context. Similarly the right panel shows a two-scale image, which superimposes a fine-scale (thin-plate) texture on a large-scale binary field.

The key idea behind hidden Markov modelling is that many complex images, scenes, and phenomena can be modelled as combinations of simpler pieces. Although the details are yet to be defined, the essence of the approach is illustrated graphically in Figure 7.2.

[1] This chapter assumes an understanding of Markovianity and random fields from Chapter 6.

P. Fieguth, *Statistical Image Processing and Multidimensional Modeling*, Information Science and Statistics, DOI 10.1007/978-1-4419-7294-1_7, © Springer Science+Business Media, LLC 2011



A nonstationary field A two-scale problem

Fig. 7.1. Two examples of complex, nonstationary behaviour. How do we construct models for such fields? Are these even Markov?

7.1 Hidden Markov Models

Hidden Markov Models (HMMs) [26, 127, 205–207, 265] have a long history in stochastic signal processing, most significantly in the speech analysis literature. The attractiveness of HMMs stems from their intuitiveness and tractability in formulating complex inverse problems.

We start with a single random field Z, as shown in the top panel of Figure 7.3. For modelling simplicity, we assert that Z is stationary, Markov, and has a local neighbourhood. The local neighbourhood must therefore be capable of decoupling a pixel Z_i from the rest of the domain, therefore it is not possible for Z to have structure on more than one scale, since a local neighbourhood cannot simultaneously discriminate two separate scales.

To induce more complex structure, we therefore need *multiple* random fields, combined in some way. The combining will be developed through the following four examples.

7.1.1 Image Denoising

If we have a Markov field $\underline{z}$ and add noise $\underline{v}$ to it, the observed sum

$$\underline{m} = \underline{z} + \underline{v} \qquad \underline{z} \sim \text{Markov} \qquad \underline{v} \sim \text{White} \tag{7.2}$$

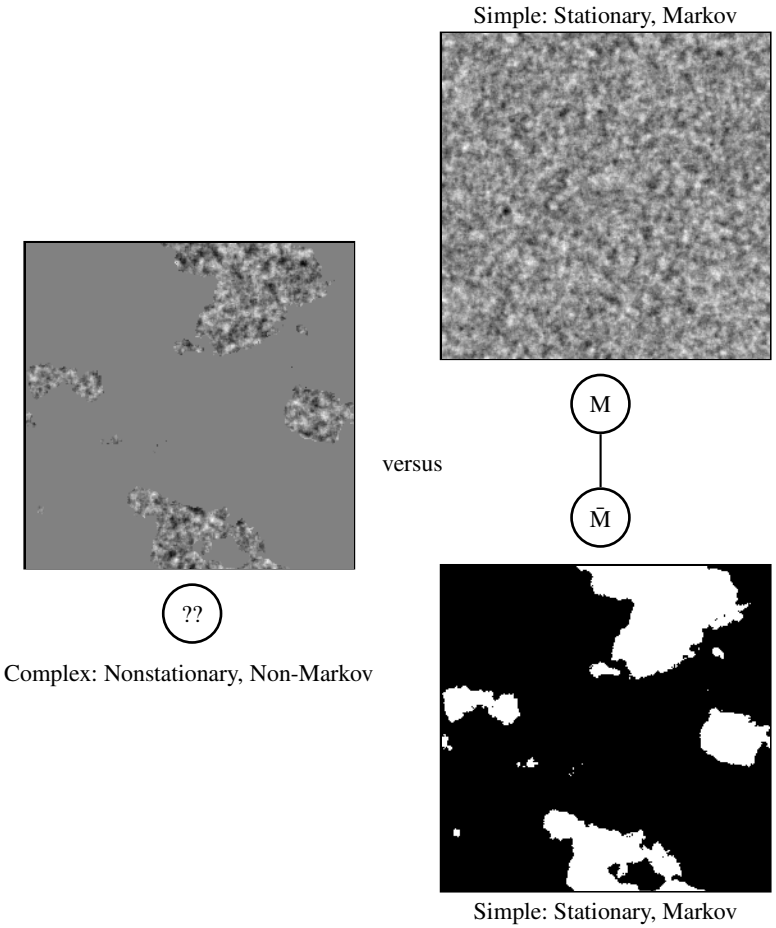
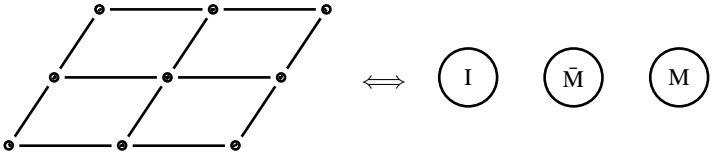


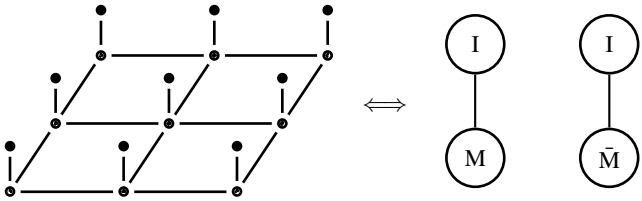
Fig. 7.2. The essential premise of hidden modelling is to decompose a complex problem, left, into multiple simple pieces, right. The non-Markovian statistics, left, become Markovian by conditioning on a hidden or underlying field.

falls directly into the context of linear inverse problems discussed throughout Chapters 2 and 3.

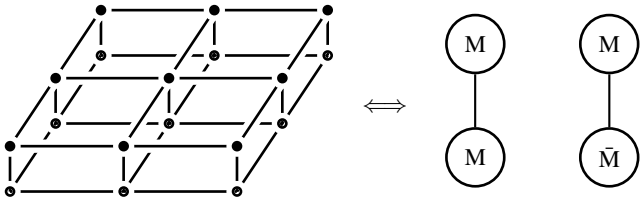
Although we know the analytical solution to (7.2), let us deliberately reformulate the problem in terms of a conditional density:



I. A single field, left, may be independent, discrete-state Markov, or continuous-state Markov, respectively.



II. Given the hidden field (○), the observed field (●) is *conditionally* independent.



III. A generalization of case II, such that the visible, conditional field is Markov, rather than independent.

Fig. 7.3. A progression in complexity of three modelling structures, from a single field (I), to a conditionally independent field (II), to a conditionally Markov field (III). The random field meshes are sketched in two dimensions, but apply equally well to one- or three-dimensional processes. Since such sketches become cumbersome for large or complex problems, we summarize each structure with a simplified diagram, right, where nodes are conditional on all connected nodes from below.

$$p(\underline{m}, \underline{z}) = p(\underline{m}|\underline{z}) \cdot p(\underline{z}) \tag{7.3}$$

$$= \underbrace{\left(\prod_i p(\underline{m}_i|\underline{z}_i)\right)}_{\text{Independent}} \cdot \underbrace{p(\underline{z})}_{\text{Markov}} \ . \tag{7.4}$$

That is, we have an underlying hidden Markov field $\underline{z}$, which makes the observed field $\underline{m}$ conditionally independent, as sketched in the middle of Figure 7.3.

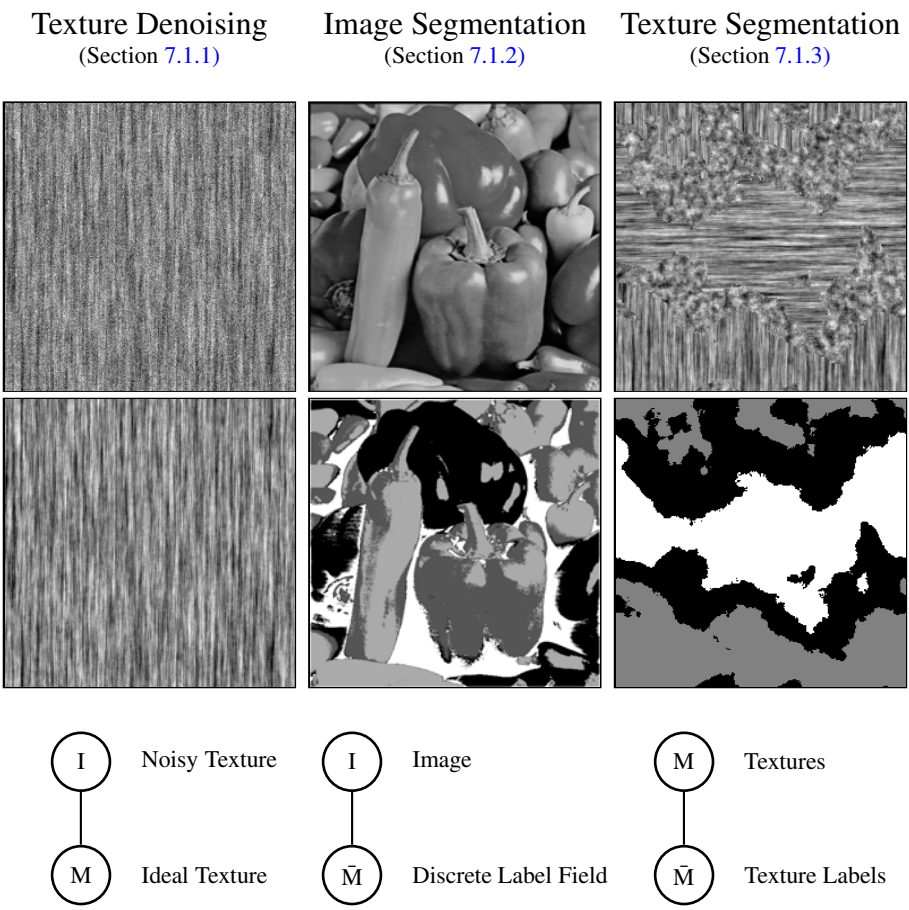


Fig. 7.4. Three illustrations of two-layer hidden Markov models. In each case, a more complex scene (top) is simplified by absorbing one component of its structure into a hidden layer (bottom). Conditioned on the hidden layer, the top image becomes independent (left, centre) or Markov (right).

7.1.2 Image Segmentation

In image segmentation [35, 36, 54, 205], we wish to take a given, possibly noisy, image M and partition it into piecewise-constant regions. In practice, segmentation may involve colour or multispectral images, or the segmentation of multiple textures (Figure 7.4); however, for the purpose of this example, let us assume M to be a noisy, piecewise-constant greyscale image, as illustrated in Figure 7.4.

The underlying (hidden) label field U is now discrete, such that $u_i = j \in \Psi$ asserts that pixel m_i belongs to the jth region. The forward model is very similar to (7.2) for denoising:

$$\underline{m} = f(\underline{u}) + \underline{v} \qquad \underline{u} \sim |\Psi|\text{-state Markov} \qquad \underline{v} \sim \text{White} \tag{7.5}$$

such that $f(j)$ identifies the grey-shade of region j. The conditional formulation proceeds as before:

$$p(\underline{m}, \underline{u}) = \underbrace{\left(\prod_i p(\underline{m}_i|\underline{u}_i)\right)}_{\text{Independent}} \cdot \underbrace{p(\underline{u})}_{\text{Markov}} . \tag{7.6}$$

Because U is now a discrete-state field, we need to select an appropriate discrete prior model, a few of which are discussed in Section 7.4.

The strength of this approach is that we have explicitly separate prior models for the visible image $p(m|u)$ and for the quite different behaviour of the underlying label field $p(u)$.

One obvious limitation to (7.6) is that the image model is simplistic, with additive white noise on piecewise-constant regions. In practice we are interested in segmenting more subtle behaviour, such as different texture models, addressed next in Section 7.1.3.

A second limitation is that the association $f()$ between region label and image value is assumed to be known ahead of time. In practice this association is found via a clustering method, such as K-means [91]. An alternative approach, based on a hidden edge model, is discussed in Section 7.1.4.

7.1.3 Texture Segmentation

We are now given an image M composed of multiple textures, as shown in Figure 7.1, which we would like to segment or decompose [225, 226, 260]. M consists of K Markov models, where the choice of model is determined by the K-state hidden field U,

$$u_i \in \Psi = \{1, \ldots, K\} \tag{7.7}$$

as illustrated in Figure 7.4.

The reader will find the generalization of (7.6) to the texture segmentation context quite straightforward by now:

$$p(\underline{m}, \underline{u}) = \underbrace{p(\underline{m}|\underline{u})}_{\text{Markov}} \cdot \underbrace{p(\underline{u})}_{\text{Markov}} . \tag{7.8}$$

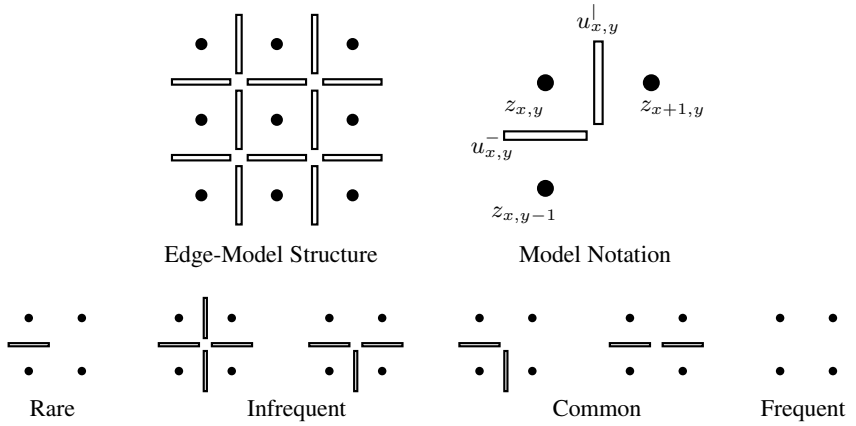


Fig. 7.5. A possible hidden model for edge detection. A hidden layer consists of binary edge elements u (rectangles, top), declaring whether or not two adjacent pixels (●) in z are separated. The hidden layer u requires a prior model, bottom, where edge terminations and junctions are rare, relative to continuing segments or no edge at all.

In the case where the observed image is noisy, an additional layer is added, such that we have hidden layers of label U and denoised texture Z:

$$p(\underline{m},\underline{z},\underline{u}) = \underbrace{\left(\prod_i p(\underline{m}_i|\underline{z}_i)\right)}_{\text{Independent}} \cdot \underbrace{p(\underline{z}|\underline{u})}_{\text{Markov}} \cdot \underbrace{p(\underline{u})}_{\text{Markov}} . \tag{7.9}$$

Indeed, the attractiveness of the hidden Markov approach is how simple and intuitive such decompositions can become.

7.1.4 Edge Detection

Suppose we wish to segment an image, as in Section 7.1.2, but without assuming knowledge of the relationship $f()$ in (7.5) between the image and the hidden labels.

The alternative to a hidden label, identifying what region a pixel belongs to, is a hidden binary state $u^{|,-}_{x,y}$, lying between adjacent pixels in z, declaring whether a region boundary is present [127] as sketched in Figure 7.5:

$$\begin{aligned} u^{|}_{x,y} = 0 \Rightarrow \text{ No edge is present } \Rightarrow z_{x,y}, z_{x+1,y} \text{ constrained to be similar.} \\ u^{|}_{x,y} = 1 \Rightarrow \text{ An edge is present } \Rightarrow z_{x,y}, z_{x+1,y} \text{ decoupled, no constraint.} \end{aligned} \tag{7.10}$$

Explicit two-dimensional indices are used here, to simplify the notation relating u and z.

If the observed image is noisy, then we have the familiar three-layer hidden Markov model

$$p(\underline{m}, \underline{z}, \underline{u}) = \underbrace{\left(\prod_{x,y} p(m_{x,y}|z_{x,y})\right)}_{\text{Independent}} \cdot \underbrace{p(\underline{z}|\underline{u})}_{\text{Markov}} \cdot \underbrace{p(\underline{u})}_{\text{2-Markov}} \quad . \tag{7.11}$$

The model requires that we specify a prior $p(u)$ for the binary edge field. In the absence of other knowledge, this prior is most easily specified via the probabilities of local edge groups, as shown in Figure 7.5. The resulting model is most easily written as a Gibbs energy (see Section 6.5):

$$\begin{aligned} H(\underline{m}, \underline{z}, \underline{u}) = & \underbrace{\sum_{x,y} (m_{x,y} - z_{x,y})^2}_{\text{Measurement Noise}} + \underbrace{\sum_{x,y} \left(1 - u^{|}_{x,y}\right)(z_{x,y} - z_{x+1,y})^2}_{\text{Vertical Edges}} \\ & + \underbrace{\sum_{x,y} \left(1 - u^{-}_{x,y}\right)(z_{x,y} - z_{x,y+1})^2}_{\text{Horizontal Edges}} \\ & + \underbrace{\sum_{x,y} H(u^{|}_{x,y}, u^{-}_{x,y}, u^{-}_{x+1,y}, u^{|}_{x,y+1})}_{\text{Edge Prior}} . \end{aligned} \tag{7.12}$$

In practice, (7.12) may be a bit simplistic, and we may instead wish to assert non-quadratic penalties between pixels, and to weight differently the various parts of the energy function.

7.2 Classes of Joint Markov Models

A very wide variety of hidden Markov models has been proposed for image modelling and analysis, as summarized in Figure 7.6. The actual labels and categories are relatively unimportant: when faced with a spatial modelling problem, no attempt should be made to fit the problem into one of these predefined categories. It is far preferable to examine the hidden behaviour present in a problem, and to propose a structure reflecting the behaviour which is actually present.

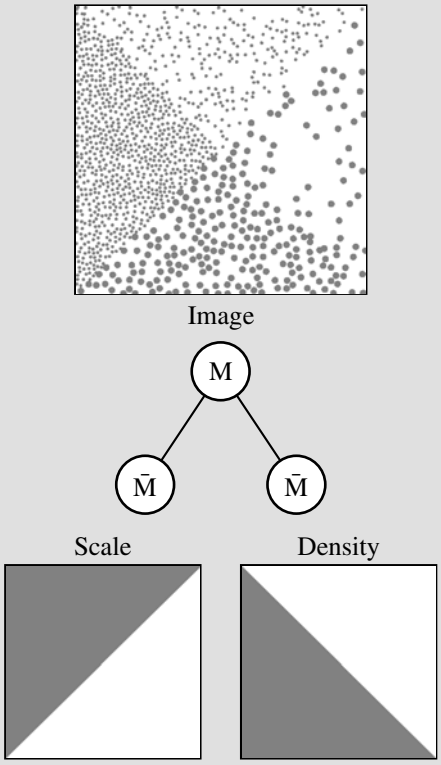
The quick survey presented here is primarily to expose the reader to the commonly employed models, with citations to the literature for further reading:

- *Markov Models* [86, 127] are discussed in detail in Chapter 6. In particular, a single, stationary Gauss–Markov random field is described by a kernel, and is able to model textures and patterns having a single characteristic scale.

Example 7.1: Multiple Hidden Fields

There is no significant distinction between having one or multiple hidden fields: for each distinct attribute or behaviour appearing in an image, we require a separate hidden field to control the attribute's presence or absence.

For example, the four regions in the relatively complex image below are described in terms of two attributes: circle size and circle density. We can therefore construct a hidden system consisting of two discrete-state hidden fields. The top image is Markov only when conditioned on *both* of the hidden fields.



Clearly an additional attribute, such as colour, would just lead to an additional hidden field.

Similarly if the attributes take on more than two possibilities, such as having circles of three different sizes, then the *scale* hidden field becomes ternary, rather than binary.

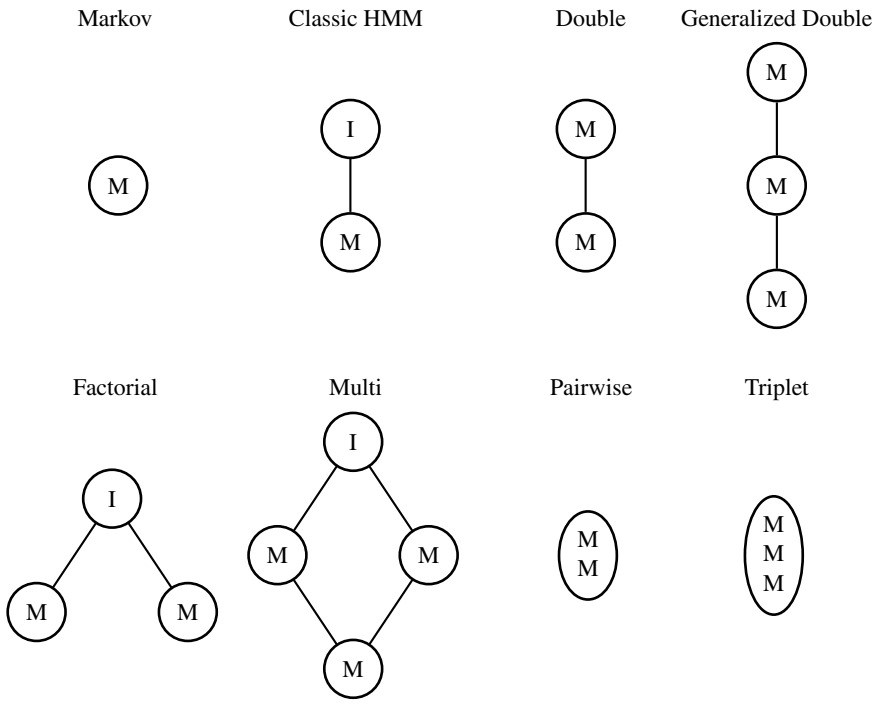


Fig. 7.6. A representative sample of the common hidden Markov models proposed for image analysis and modelling. The difference between the *Double* and *Pairwise* models is that the two fields in the *Double* model are Markov and conditionally Markov, whereas in the *Pairwise* model the fields are jointly Markov, but neither field is Markov on its own.

- *Hidden Markov Models* [26, 127, 205–207] have an underlying Markov field Z, such that the visible field $M|Z$ is conditionally independent. This narrow interpretation of a hidden Markov model essentially applies only to noisy Markov fields, so for most problems of multidimensional modelling more substantial structures are required.

- The *Double Model* [230] allows the conditional field to be Markov, allowing for hidden labels and visible textures, as in Section 7.1.3. The model can clearly be generalized to allow additional layers.

- The *Factorial Model* [193] allows for a greater number of conditioning fields, as in Example 7.1, where a given field depends on multiple factors, each of which may be independently modelled as a random field.

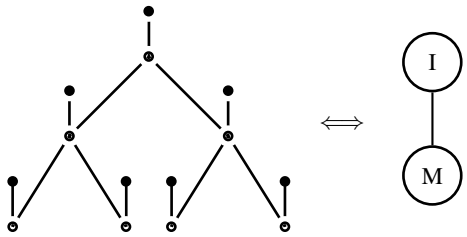


Fig. 7.7. A hidden Markov tree: The hidden binary state under each wavelet coefficient declares whether the corresponding coefficient is of small or large variance. Although the structure appears relatively complex, it is actually acyclic (no loops), so inference and estimation can be done very efficiently.

- The joint models, such as the *Pairwise* [260] and *Triplet* [21] models, give increased model flexibility but at a substantial computational cost. A pair of fields (Z, U) being jointly Markov implies that $(Z|U), (U|Z)$ are both conditionally Markov, however neither Z nor U is Markov on its own.

  Similarly a triple (M, Z, U) being jointly Markov implies that doubly-conditioned fields $(M|Z, U)$ are conditionally Markov, but singly-conditioned $(Z|U), (U|Z)$ and unconditioned Z, U are *not*.

- A great many specialized models have been proposed, including the edge model [127] of Section 7.1.4, multimodels [22], and wavelet models [77,274]. As multidimensional spatial operators, wavelets (discussed in Chapter 8) are of particular interest. The hidden Markov tree, illustrated in Figure 7.7, uses a binary hidden state to model the observation that the variance of child coefficients tends to match the variance of the parent.

- Spatial hidden Markov models are part of a much larger literature on complex and distributed Bayesian inference, including graphical models [180], Bayesian networks [177,249], and hierarchical hidden Markov models [116].

7.3 Conditional Random Fields

The models we have seen thus far are *generative*: a joint distribution

$$p(Z, U) = p(Z|U)p(U) \tag{7.13}$$

describes all of the statistics of Z, meaning that we can generate or draw samples of Z.

Although the explicit goal of this text is the development of multidimensional generative models, let us briefly consider a different perspective, which is that for certain problems the generative model says much more than we need. Particularly for labelling or segmentation problems, if we are given an image M from which we wish to infer labels U, we do *not* need to be able to describe the statistics of M. M is given; we only need to be able to describe the dependence

$$p(U|M) \tag{7.14}$$

of U on M. This is a conditional or *discriminative* model, able to discriminate between possible label fields $U_1, U_2, \ldots$ but without any implied model for M.

The generative models of this chapter, such as in (7.6), were simplified by assuming the conditioned distribution to be Markov or independent. Consider the following:

$$\underbrace{p(M|U) \to \prod_i p(m_i|U)}_{\text{Generative}} \qquad \underbrace{p(U|M) \to \prod_i p(u_i|M)}_{\text{Discriminative}}. \tag{7.15}$$

We observe:

- The generative model describes a single image pixel m_i on the basis of a nonlocal set of labels,
- The discriminative model describes a single label u_i on the basis of a nonlocal portion of the image.

So in direct contrast to the generative approach, discriminative methods are attractive because the image M is not modelled, typically leading to models having fewer parameters and increased robustness, and because the single label u_i is conditioned or described in terms of *nonlocal* portions of the image. The only aspect which is missing is some form of interdependence between label values, so we would prefer a discriminative model along the lines of

$$p(U|M) \to \prod_i p(u_i|u_{\mathcal{N}_i}, M) \tag{7.16}$$

over some neighbourhood $\mathcal{N}_i$.

One approach to discriminative modelling is that of conditional random fields [198, 295], in which the conditional distribution is written as a Gibbs model (Section 6.5)

$$p(U|M) = \frac{1}{\mathbb{Z}} \exp\left(\sum_i f_1(u_i, M) + f_2(u_i, u_{i-1}, M)\right). \tag{7.17}$$

The labels $\{u_i\}$ are assumed to be sequentially ordered, and the dependence of U on M is described by feature functions f_1 and f_2, the single-site and transition features, respectively.

It is, of course, possible to generalize conditional random fields to multidimensional problems. We have already repeatedly encountered this question, in the context of coupling and complexity in Section 5.2, and in the context of one-dimensional versus multidimensional random fields in Chapter 6. The short summary is that sequential models are efficient but tend to lead to poorer models, so non-sequential iterated approaches are preferred, based on some sort of neighbourhood, as in

$$p(U|M) = \frac{1}{\mathbb{Z}} \exp\left(\sum_i f_1(u_i, M) + f_2(u_i, u_{\mathcal{N}_i}, M)\right). \tag{7.18}$$

In addition to not modelling M, the great strength of the conditional random field approach is the use of a Gibbs framework, which allows great flexibility in modelling, since nonlinear/discrete-state models are easily accommodated and there are no issues of normalization or positive-definiteness. For example, the image segmentation method in [157] is based on a conditional random field model

$$p(U|M) = \frac{1}{\mathbb{Z}} \exp\left(\sum_i f_1(u_i, M, \underline{\theta}) + f_2(u_i, u_{\mathcal{N}_i^{\text{Local}}}, \underline{\theta}) + f_3(u_i, u_{\mathcal{N}_i^{\text{Non-Local}}}, \underline{\theta})\right) \tag{7.19}$$

with unknown model parameters $\underline{\theta}$ to be learned. Conditional random fields have seen growing application in computer vision, to problems of labelling, segmentation, and object tracking [157, 196, 327].

7.4 Discrete-State Models

In many cases the hidden field is a discrete label, such as the high/low variance of a wavelet coefficient, the region in an image, the identity of a texture, or the classification of a pattern. In any such case, the hidden field will require a discrete-state model. Although most discrete-state models are fairly simple and intuitive, there are two key difficulties:

1. There are surprisingly few models to choose from, and of these even fewer that actually provide anything but the most trivial structure.
2. As soon as a discrete-state model is included, the overall problem becomes, by definition, a discontinuous function of the state, and therefore nonlinear.

It is because of nonlinearity that the majority of this text has focused on the continuous-state case, nevertheless the importance of hidden Markov models motivates at least an overview of discrete-state methods.

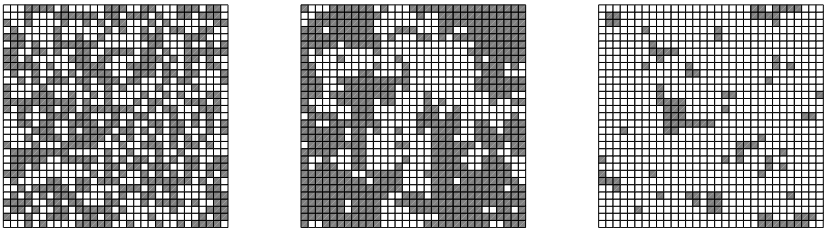


Fig. 7.8. The two-dimensional Ising model: three random samples are shown, for different values of inverse-temperature parameter β. As β is increased, the coupling between pixels is more strongly asserted, and the prior states a greater preference for homogeneous regions.

7.4.1 Local Gibbs Models

The *Ising* model [26, 171, 335] is the most famous of all discrete-state models. Originally developed as a model of ferromagnetism in statistical mechanics, its early fame stemmed from the fact that the two-dimensional problem was solved, analytically, by Onsager in 1944.

The regular Ising model, with no external field, is a simple, first-order Markov model:

$$H(U) = -\sum_{i,j}\left(u_{i,j}u_{i+1,j} + u_{i,j}u_{i,j+1}\right) \quad u_{i,j} \in \{-1, 1\} = \Psi. \tag{7.20}$$

Recall from (6.39) in Chapter 6 the relationship between energy and probability density for Gibbs random fields:

$$p(\underline{u}) = \frac{1}{\mathbb{Z}} e^{-\beta H(\underline{u})}. \tag{7.21}$$

Thus a lower (more negative) energy H is associated with a higher probability density; that is, the Ising model states a preference for regions of constant state value in Z.

Three random samples of $\underline{u}$, drawn from $p(\underline{u})$ in (7.21), are shown in Figure 7.8. The Ising prior is a function of only a single parameter, the inverse-temperature β in (7.21). Correspondingly, the Ising model is a very weak prior, capable only of stating a vague preference for large blob-like objects.

The *Potts* model [26, 258, 335] is the generalization of the Ising model to the K-ary case, for states taking on one of $K > 2$ discrete values:

$$H(U) = -\sum_{i,j}\delta_{u_{i,j},u_{i,j-1}} - \sum_{i,j}\delta_{u_{i,j},u_{i-1,j}} \quad u_{i,j} \in \{1, 2, \ldots, K\} = \Psi. \tag{7.22}$$

Qualitatively, the behaviour of the Potts model is very similar to that of Ising — it is a single-parameter first-order local prior, preferring large homogeneous regions.

7.4.2 Nonlocal Statistical-Target Models

The strength of the Gibbs approach to statistical modelling lies in the assertion of intuitive energy functions. That is, unlike covariance matrices (which need to satisfy subtle positive-definiteness requirements), the energy function has only one requirement:

Assign lower values to "better" states, and larger values to "poorer" states.

One of the simplest and most common approaches is to define one or more attributes $A_q(U)$, with idealized target values learned from training data

$$T_q = A_q(\check{U}) \tag{7.23}$$

such that the energy function is then expressed as the degree of inconsistency between the attributes of a given state Z and the target:

$$H(U) = \|\underline{T} - \underline{A}(U)\|. \tag{7.24}$$

In the simplest case, we just penalize the squared difference between attribute and target:

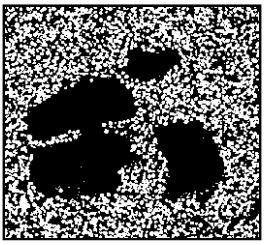
$$H(U) = \sum_q \big(T_q - A_q(U)\big)^2. \tag{7.25}$$

Any feature which can be written as a function of a discrete-state field can therefore form the basis for a Gibbs energy. Common features include the presence in U of horizontal, vertical, or diagonal lines of length q, the two-point correlation as a function of offset q, the fraction of states having a given state value, or many other morphological features.

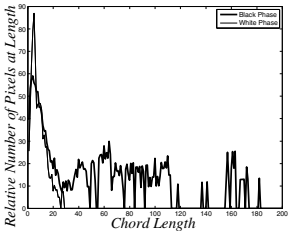
The *Chordlength* model [312] characterizes a random field on the basis of the distribution of black or white chords. Define $A_l^{d,v}(U)$ to be the number of chords present in U of length l and state value v in direction d. Then the energy comparing a given field U to some ground truth $\check{U}$ is

$$H(U) = \sum_{d,v,l} \big(A_l^{d,v}(U) - A_l^{d,v}(\check{U})\big)^2. \tag{7.26}$$

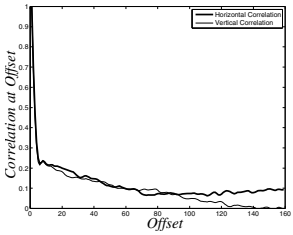
Related is the *Correlation* model [312], which characterizes a random field based on autocorrelation. Clearly a variety of alternatives is present, such as whether the correlation is specified separately in the horizontal and vertical directions, or a single isotropic model, and also the maximum offset to which the correlation is asserted.



Binary Random Field



Target Chordlength



Target Correlation

Fig. 7.9. Two possible energy targets for a binary random field: chordlengths (middle) and correlation (right). Observe how the model reveals the presence of large black pores via a heightened target on long black chords in the middle panel.

Because the extracted attributes $A()$ are nonlocal in support, both the correlation and chordlength are *global* models, distinctly different from the local Markov models seen throughout Chapter 6. Figure 7.9 illustrates extracted target distributions for a sample, binary random field of a porous medium.

7.4.3 Local Joint Models

The strength of the chordlength and correlation models is that they are global and are able to describe large-scale phenomena, such as the large black pores seen in Figure 7.9. On the other hand, the weakness of these nonlocal models is that they are poor at capturing and describing local, fine-scale, detailed morphology.

As an alternative, local joint models over a small window of state elements have been proposed. Although *Local Binary Patterns* (LBP) [241, 242] have been used most frequently as features in pattern recognition problems, such as texture classification, as image attributes they are equally applicable to being asserted in an energy function (7.25). In their most basic form, LBP features are based on the eight pixels surrounding a central one, are rotation invariant, and are based on those nine patterns, shown in Figure 7.10, empirically deemed to be the most significant.

Very similar is the *Local Histogram* model [5], which preserves the joint probability of each of the 2^9 configurations of binary elements in a 3×3 window, leading to energy

$$H(U) = \sum_{q=0}^{2^9-1} \frac{\left(P_q(\check{U}) - P_q(U)\right)^2}{\sigma_q + \epsilon}, \tag{7.27}$$

where σ_q represents the variability, from sample to sample, of the probability P_q of configuration q, and small $\epsilon > 0$ is chosen to avoid a division by zero for impossible

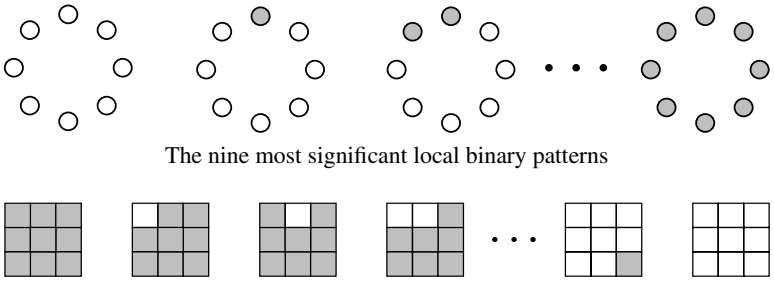


Fig. 7.10. In contrast to the non-local Gibbs models of Figure 7.9, here two local models are proposed, based on Local Binary Patterns (top), or an exhaustive set of joint combinations (bottom). If a prior model with substantial structure and morphology is required, these local models are best combined with nonlocal ones, or used in a hierarchical setting.

(zero probability) configurations. In practice, the number of configurations may be reduced from 2^9 by asserting reflection / rotation invariance.

The flexibility of the Gibbs energy approach allows new energy functions to be proposed at will. For example, because the LBP and histogram models are local, there may be some merit to combining them with a nonlocal model such as chordlength:

$$H_{\text{Combined}}(U) = H_{\text{Chordlength}}(U) + \alpha H_{\text{Histogram}}(U). \tag{7.28}$$

7.5 Model Determination

Whereas Section 5.7 examined the question of modelling, in general, and Section 6.6 in the context of Markov fields, here we consider modelling issues for hidden fields.

If there is only a single, fixed prior model for the hidden field, then given a model and ground-truth data for U we can infer the model parameters $\underline{\theta}$, as was done in Figure 7.9.

On the other hand, if the model parameters $\underline{\theta}$ may vary with and be a function of image Z, then $\underline{\theta}$ needs to be learned in each context. Furthermore, since the hidden field is, well, hidden, it is not available for parameter learning, and $\underline{\theta}$ must be learned from the visible field Z or its measurements M.

The difficulty, as discussed in Section 5.7, is that without knowing U, it is exceptionally difficult to formulate the maximum-likelihood estimation problem

$$\hat{\underline{\theta}} = \arg_{\underline{\theta}} \max p(Z|\underline{\theta}) \qquad \text{or} \qquad \hat{\underline{\theta}} = \arg_{\underline{\theta}} \max p(M|\underline{\theta}) \tag{7.29}$$

Algorithm 2 Simplified Expectation Maximization

Goals: Estimate the parameters $\underline{\theta}$ for hidden field U
Function $[\hat{U}, \hat{\underline{\theta}}] = \mathbf{EM}(Z)$
  Initialize $\hat{\underline{\theta}}$
  while not converged **do**
    E-Step: Given model parameters $\hat{\underline{\theta}}$, compute the hidden estimates $\hat{U} \leftarrow E[U|Z, \hat{\underline{\theta}}]$.
    M-Step: Given hidden estimates $\hat{U}$, compute ML estimates $\hat{\underline{\theta}} \leftarrow \arg_{\underline{\theta}} \max p(\hat{U}|\underline{\theta})$.
  end while

since the influence of $\underline{\theta}$ on Z, M is felt *via* U, whereas the hidden maximum-likelihood problem

$$\hat{\underline{\theta}} = \arg_{\underline{\theta}} \max p(U|\underline{\theta}) \tag{7.30}$$

is normally very easy, if U were known. The basic idea, then, is a straightforward extension of the above insight, which is to alternate between estimating $\underline{\theta}$ and U, as shown in Algorithm 2. This is a somewhat simplified description of what is known as the EM (Expectation–Maximization) algorithm [25,85,275], which is near-universally used in hidden Markov models.[2] Slightly more precise than Algorithm 2, the EM algorithm consists of two steps:

E-STEP: Compute the expectation $E_{\underline{u}}\left[p\left(\underline{z}, \underline{u}|\underline{\theta}\right)|\underline{z}, \underline{\theta}_i\right]$

M-STEP: Find $\underline{\theta}_{i+1}$ to maximize the expectation $E_{\underline{u}}$

The EM algorithm is nothing more than the iteration of these two steps; EM does not tell you *how* to compute these steps. Indeed, given that the E-Step is essentially an estimation problem, implementing the EM method for a large, multidimensional problem will rely on the estimation methods of this text to compute the estimates. The latter M-Step is normally a comparatively straightforward ML problem.

Although the EM algorithm cannot guarantee convergence of $\hat{\underline{\theta}}$ to the ML estimate, the likelihood of $\hat{\underline{\theta}}$ is guaranteed not to decrease with any iteration, meaning that EM is guaranteed to reach a local maximum in likelihood space.

For acyclic HMMs, in which the statistical dependencies have no loops (Figure 5.1), efficient EM implementations exist based on the forward–backward algorithm [265, 275] for sequential problems, and a very similar upward–downward algorithm on trees [77, 275].


Fig. 7.11. Two standard test images, "Peppers" and "House". The images are plotted in greyscale, however the underlying images are actually in colour.

Application 7: Image Segmentation

Let us continue to explore the image segmentation problem, as sketched in Figure 7.4:

Given an image I, we wish to partition the image into K segments.

Because each image pixel $I_{x,y}$ is placed into one of K discrete segments, we clearly have a hidden label field $U_{x,y} \in \{1, \ldots, K\} = \Psi$, describing the partition of each associated pixel.

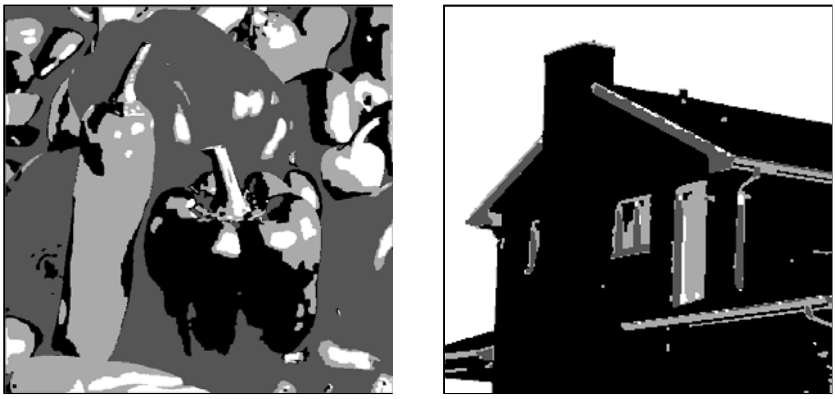
The goal of this application is to illustrate how the prior model for U affects the segmentation result. We will demonstrate segmentation on the two images of Figure 7.11.

No Prior Model

We begin with no prior model for U, such that nothing is known about the spatial properties of U, and the segmentation problem is non-Bayesian. To group the elements of I into one of K groups, on the basis of the colour of each pixel, is known as a *clustering* problem:

[2] In the context of hidden Markov models the EM algorithm is also known as the Baum–Welch algorithm.

K-means segmentation



K-means + Potts prior segmentation

Fig. 7.12. Image segmentation based on a global K-means clustering, with (bottom) and without (top) a spatial prior model on the hidden label field.

$$[\hat{\underline{c}}, \hat{U}] = \arg_{c,U} \min \sum_{x,y} \|I_{x,y} - c_{U_{x,y}}\|. \tag{7.31}$$

A very standard solution to clustering is the K-means algorithm [91], for which the segmentation is shown in Figure 7.12. Because each cluster center c_i is a *global* parameter, all parts of the image are segmented consistently. However, the absence of any spatial prior means that adjacent pixels are in no way constrained, and so we see the creation of large numbers of single-pixel regions at the interface between two segments.

K-Means with Potts Prior

Given the K-means constraint of (7.31), it is straightforward to modify the criterion to include a spatial constraint, such as a Potts prior, on the hidden label field L:

$$\hat{U} = \arg_U \min \sum_{x,y} \|I_{x,y} - c_{U_{x,y}}\| + \text{Potts}(U) \tag{7.32}$$

$$= \arg_U \min \sum_{x,y} \|I_{x,y} - c_{U_{x,y}}\| + \sum_{x,y} \delta_{U_{x,y},U_{x,y-1}} + \delta_{U_{x,y},U_{x-1,y}} \tag{7.33}$$

where the Potts prior is taken from (7.22).

Adding such a prior asserts a preference for spatially homogeneous regions, with the consequence that single-pixel regions are eliminated and leading to a clean, robust segmentation, as seen in Figure 7.12.

Potts Prior

The K-means approaches of Figure 7.12 have a *global* criterion, in that a single set of target colours $\{c_i\}$ applies to the entire image. This requires, however, that the image be well described in terms of a fixed set of colours and that K be known. Counterexamples would be an image having regions of smoothly varying colour, or an image with multiple regions of very similar colour which are meant to be separately segmented.

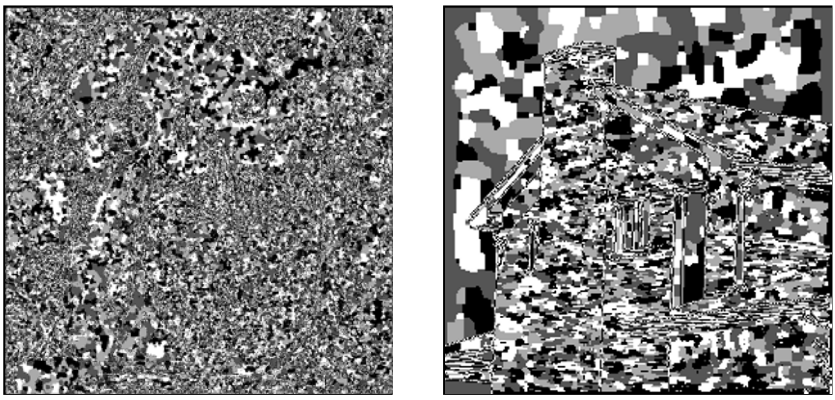
As an alternative we can consider a *local* model, such that the criterion is written entirely in terms of the local behaviour of both image and hidden layer. For example, we can assert a Potts prior, and penalizing the difference in adjacent image pixels if they belong to the same region:

$$\begin{aligned}\hat{U} = \arg_U \min \sum_{x,y} & \delta_{U_{x,y},U_{x,y-1}} \big(\|I_{x,y} - I_{x,y-1}\| - \gamma\big) + \\ & \delta_{U_{x,y},U_{x-1,y}} \big(\|I_{x,y} - I_{x-1,y}\| - \gamma\big) \quad U_{x,y} \in \Psi,\end{aligned} \tag{7.34}$$

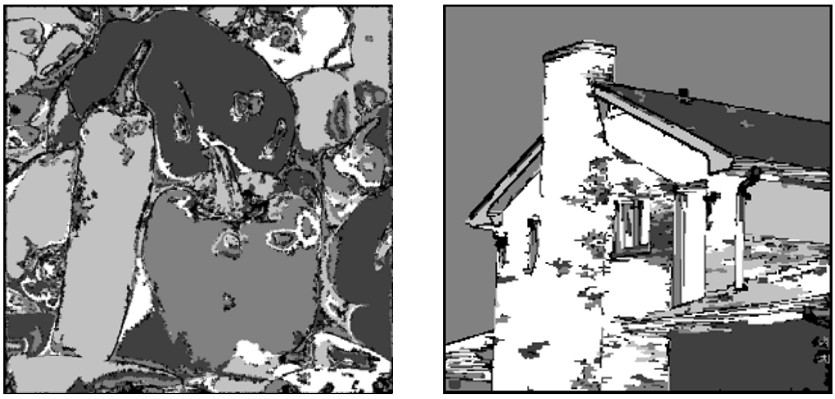
where constant γ controls the degree to which the Potts prior is asserted.

Although K must still be specified, because of the locality of the model, K no longer limits the total number of distinct segments, as in (7.33). Rather, K now asserts how many different regions may touch; a larger K gives more flexibility to the model, but also (typically) increases complexity.

As discussed further in Chapter 8, because the model is local it is difficult to form large regions: the number of iterations needed is quadratic in region size or extent. That is, for an $N \times N$ image, with regions a fixed fraction of the height or width of the image, the total complexity is



Potts Prior



Potts Prior, Repeatedly Aggregated

Fig. 7.13. Image segmentation using only *local* models, in contrast to the *global* constraints imposed in Figure 7.12. Because single pixels are flipped, one at a time, it is exceptionally difficult to converge to large homogeneous regions, top. By flipping *regions*, rather than *pixels*, much larger regions can readily be formed.

$$\text{Complexity per Iteration} \cdot \text{Number of Iterations} = \mathcal{O}\big(N^2 K\big) \cdot \mathcal{O}\big(N^2\big) = \mathcal{O}\big(K N^4\big). \tag{7.35}$$

The top panel in Figure 7.13 makes this clear: after more than 100 iterations, there are a great many small regions present. The formation of large regions is exceptionally slow, even in very simple, smooth regions such as the sky of the "House" image.

It is possible to take these large numbers of small regions and create metrics [217] for grouping regions. Similarly we can treat each region as a single hidden element, such that our energy function is written in terms of regions [329], rather than pixels,

such that (7.34) becomes

$$\hat{U} = \arg_U \min \sum_i \sum_{j \in \mathcal{N}_i} \delta_{U_i, U_j} \left(\| I_i - I_j \| - \gamma \right) \qquad U_i \in \Psi, \tag{7.36}$$

where $\mathcal{N}_j$ represents the set of regions adjacent to region j.

By doing such grouping repeatedly, as discussed in Section 8.6, the overall convergence is accelerated greatly, leading to the segmentation in the bottom panel of Figure 7.13. Because of the absence of a global criterion, we do see many more local regions here, relative to the global segmentation in Figure 7.12.

For Further Study

The Viterbi method of Section 4.5.1 generates the estimated state sequence for hidden Markov chains; generalizations to hidden random fields are discussed in Chapter 11.

The paper by Geman & Geman [127] is strongly recommended, as understanding their edge model, discussed in Section 7.1.4, will leave the reader with a good understanding of the use of hidden models with random fields.

The text by Li and Gray [205] is recommended for readers wishing to understand hidden models more deeply.

The Baum–Welch algorithm [19] is widely used to learn the parameters in HMMs. For discrete-state HMMs, the Forward–Backward algorithm [52, 96, 205] is a standard, iterative approach for state estimation.

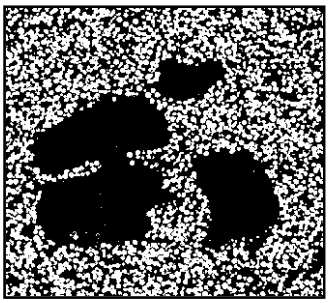
To experiment in MATLAB, readers may be interested in the Image Processing toolbox, the Hidden Markov Model toolbox, or the Conditional Random Fields toolbox.

Sample Problems

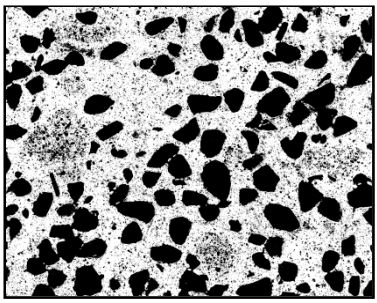
Problem 7.1: Hidden Models for Simple Binary Fields

For the two-phase random field in Figure 7.14, qualitatively sketch the associated hidden model along the lines of what is shown in Example 7.1:

(a) What sort of visual behaviour does the hidden state describe?
(b) Is the hidden state discrete or continuous? If discrete, what order is it (binary, ternary etc.)?



(See Problem 7.1)



(See Problem 7.2)

Fig. 7.14. Two sample binary random fields. A two-phase example, left, and a more complex structure, right.

(c) Is the hidden state Markov? Why / why not?
(d) Is the visible state conditionally Markov or conditionally independent?

Problem 7.2: Hidden Models for Complex Binary Fields

The binary random field shown in the right panel of Figure 7.14 shows multiple sorts of structures: pores, regions of high density, and a background region of lower density. Repeat Problem 7.1 for this image.

Problem 7.3: Pairwise Random Fields

Suppose that X, Y are jointly Markov, two-dimensional, and Gaussian.

(a) Write the explicit statement of Markovianity for the joint pair (X, Y), along the lines of (6.13).
(b) Prove that $(X|Y)$ and $(Y|X)$ are Markov.

Problem 7.4: Triplet Random Fields

Suppose that X, Y, Z are jointly Markov, two-dimensional, and Gaussian.

(a) Write the explicit statement of Markovianity for the joint triplet (X, Y, Z), along the lines of (6.13).
(b) Prove that $(X|Y, Z)$ is Markov.
(c) Argue why or in what circumstances $(X|Y)$ is *not* Markov

Problem 7.5: Three-Dimensional Random Fields

(a) Describe briefly the similarities and differences between
  (i) A 3D Markov random field, and
  (ii) A hidden chain of equally-sized 2D Markov random fields, linked via conditioning, like a long version of the Generalized Double model in Figure 7.6.
(b) Prove that a chain of equally-sized 2D Markov random fields satisfies the conditions of 3D Markovianity.
(c) Suggest a problem that might be more easily represented as multiple 2D fields rather than a single 3D field.
(d) Under what circumstances might a single 3D field be more convenient than a hidden chain of 2D fields?

Problem 7.6: Open-Ended Real-Data Problem — Segmentation

There is a very large literature on image segmentation [36,54,258], with a subset specialized to the use of random fields in segmentation [35,226], and finally more specifically to the use of hidden/pairwise/triplet random fields [21,205,230,260]. Select one or more hidden-Markov segmentation methods, and apply the selected methods to standard segmentation test images.

Compare your approach with those of non-Markov methods, such as based on watershed or region-growing.

8

Changes of Basis

The previous chapters have focused on the definition of a model, and corresponding estimator or sampler, for some random vector $\underline{z}$. Explicit throughout the preceding chapters has been the assumption that $\underline{z}$ contains a set of spatial elements or image pixels; that is, that $\underline{z}$ represents the raw, underlying random field of interest.

However, a given model for $\underline{z}$ clearly implies a model for any linear function of $\underline{z}$:

$$\underline{z} \sim P \quad \xrightarrow{\bar{\underline{z}} = F\underline{z}} \quad \bar{\underline{z}} \sim FPF^T \tag{8.1}$$

where, if $F \neq I$, the elements of $\bar{\underline{z}}$ are no longer image pixels, rather some linear function, possibly local or nonlocal, of the original random field. Furthermore, if $F \neq I$ is square and invertible, then the transformation from $\underline{z}$ to $\bar{\underline{z}}$ is referred to as a *change of basis*.

Clearly what is lost in such a change is the simple, intuitive understanding of the state elements as image pixels. It is also possible that certain desirable properties of the model may be lost; in particular, most models increase in density (become less sparse) through a change of basis, and models which are stationary or Markov normally become nonstationary or non-Markov [197, 199, 254]. Nevertheless there is much to be gained in terms of numerical robustness and computational efficiency.

In particular, the poor conditioning of many estimation problems, especially in our context of random fields, stems from the locality of most Markov and deterministic constraint models (e.g., Figures 5.10 and 5.11 for deterministic kernels $\mathcal{Q}$, and Example 6.1 on page 190 for Markov kernels $\mathcal{G}$). Figure 8.1 illustrates the problem: for a local operator to assert a statistical relationship between distantly-separated pixels requires information to be passed indirectly through many state elements. If z_1 and z_6 in Figure 8.1 are uncorrelated then there is no statistical relationship to assert, and the problem may be well conditioned; however if z_1 and z_6 are very tightly constrained then we have a very strict assertion, but which needs to be inferred *implicitly* from the *repeated* application of a model. Such a tight correlation would be the case

P. Fieguth, *Statistical Image Processing and Multidimensional Modeling*, Information Science and Statistics, DOI 10.1007/978-1-4419-7294-1_8, © Springer Science+Business Media, LLC 2011

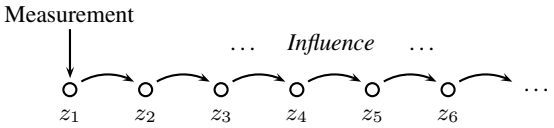


Fig. 8.1. Local models lead to ill-conditioning because of indirection: given a measurement of element z_1 and a first-order prior model, we need to infer z_2 from z_1, then z_3 from z_2 and so on. The greater the number of levels of indirection, the more poorly conditioned is the system matrix and the more slowly a corresponding iterative solver converges.

with a smooth spatial correlation, such as a Gaussian, which explains much of the conditioning behaviour in Table 5.2 (page 163).

The indirection of statistical assertions and information flow can be clearly seen in Figure 8.2, which plots the iterative solution (discussed in Chapter 9) to a linear system. At the measured locations the state elements respond strongly to the measurement, however state elements some distance away from the measurements have only barely responded, and it will require a great many iterations before converging. The key idea in this chapter is that a change of basis, coupling non-neighbouring state elements, might accelerate this convergence.

The fundamental mathematical approach to improving conditioning is to affect the distribution of eigenvalues in the system, however in most cases the details of the eigendistribution are unknown for a given inverse problem. Instead, in practice the desire to improve conditioning amounts to reducing the degree of indirect statistical assertions between the elements of $\underline{z}$, for which there are three basic approaches, all of which are explored in this chapter:

1. Reduce the Strength of Statistical Assertions:

   That is, find a change of basis to accomplish some degree of state decorrelation. The method of principal components is based on this idea.

2. Make the Problem Smaller:

   That is, find a reduction of basis, whether by subsampling or by local methods, working on subsets of the random field, that the number of state elements is reduced.

3. Make the Model Nonlocal, Reducing Indirection:

   That is, introduce a change of basis in which the basis elements are nonlocal, allowing spatially separated state elements to be coupled, reducing the number of steps of interaction.

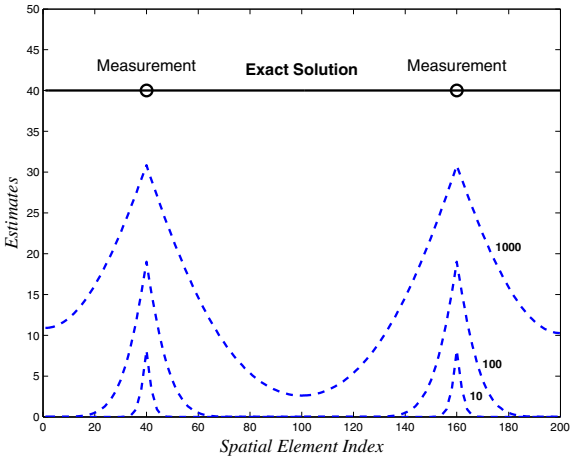


Fig. 8.2. We can see the indirect nature of the local system, discussed in Figure 8.1, by examining an iterative solution to a simple, first-order interpolation problem after 10, 100, and 1000 iterations. Near the measurements the state elements are strongly pulled towards the measurement, however the information provided by the measurement decays exponentially as we move away from the measured locations.

Because of their computational benefits, changes of basis take place implicitly in many efficient algorithms, as will be seen in Chapter 9. However the focus of this chapter continues to be *modelling*: how can we use basis changes to change a problem, possibly to reduce it in size, to decouple it, or to restructure it in some fashion?

The chapter begins with an overview of the mathematics of basis changes, followed by three approaches for basis *reduction*, and finally ending with three approaches to computationally efficient basis changes.

8.1 Change of Basis

A large number of approaches to basis change are possible, however they can be divided into two general approaches:

1. *Explicit* changes, in which the entire problem, including the measurements, is transformed into a new domain. All aspects of statistical processing — modelling, estimation, prior / posterior sampling — can be undertaken in the transformed domain.

2. *Implicit* changes, in which an estimation problem is restructured such that the solution is found in the transformed domain, and then projected back into the original. Although somewhat simpler than an explicit change, implicit changes of basis apply only to estimation.

Explicit Basis Change

Consider a change of basis, as in (8.1). We transform a problem, solve it in the transformed domain, from which the solution is projected back to the original setting:

$$\begin{array}{cccc} \textit{Original Domain} & & \textit{Transformed Domain} \\ \underline{z} \sim P & \xrightarrow{\quad F \quad} & \underline{\bar{z}} \sim \bar{P} \\ \underline{m} = C\underline{z} + \underline{v} & & \underline{\bar{m}} = \bar{C}\underline{\bar{z}} + \underline{\bar{v}} \\ & & \Big\downarrow \text{Estimation} \\ \underline{\hat{z}} \sim \tilde{P} & \xleftarrow{\quad F^{-1} \quad} & \underline{\hat{\bar{z}}} \sim \tilde{\bar{P}} \end{array} \tag{8.2}$$

where the transformed problem is given by

$$\underline{\bar{z}} = F\underline{z} \sim \bar{P} = FPF^T \tag{8.3}$$

$$\underline{\bar{m}} = F\underline{m} = FCF^{-1}\underline{\bar{z}} + F\underline{v} = \bar{C}\underline{\bar{z}} + \underline{\bar{v}}. \tag{8.4}$$

That (8.2) is written as solving an estimation problem is for illustrative purposes only; really there is a wholly new model $(\bar{C}, \bar{P})$, from which prior / posterior samples could be generated, model parameters estimated, or likelihood tests performed, in addition to estimation. Three further comments are in order:

1. The purpose behind such a basis change is to make the problem easier in the transformed domain. For example, the transformed statistics FPF^T should be well conditioned and/or sparse.

2. We explicitly need both a forward F and inverse F^{-1} transform. If F alone is specified, finding the inverse transform may be extremely difficult.

3. The transformation of the measurements implicitly assumes the measurements to be dense, meaning that the number of measurements and unknowns is equal.

   An important special case, which occurs in many image processing contexts (such as denoising[1]) is where every pixel is observed, in which case $\bar{C} = C = I$.

[1] See Figure C.5 in Appendix C.3.

The key to making this change of basis practical is to select an *orthogonal* transformation for F, in which case

$\Rightarrow$ The inverse transformation is easily found: $F^{-1} = F^T$.
$\Rightarrow$ If the measurement noise $\underline{v}$ is white, then $\bar{\underline{v}}$ is also white.

These properties make orthogonal transformations one of the key tools in signal/image modelling and analysis. Indeed, later this chapter we show several extremely well-known orthogonal transformations: the Fourier transform (Section 8.3), the Wavelet transform (Section 8.4.2), and the Karhunen–Loeve transform (Section 8.2.1).

There is, in fact, a well-known orthogonal transformation which greatly simplifies statistical sampling and estimation. Consider the eigendecomposition $P = V^T \Lambda V$ of covariance P:

$$\begin{array}{ccc} \underline{z} \sim P & \xrightarrow{F = V} & \bar{\underline{z}} \sim \bar{P} = VPV^T = \Lambda \\ & & (\Lambda \text{ Diagonal}) \\ & & \Big\downarrow \text{ Trivial} \\ \hat{\underline{z}} \sim \tilde{P} & \xleftarrow{F^{-1} = V^T} & \hat{\bar{\underline{z}}} \sim \tilde{\bar{P}} \end{array} \tag{8.5}$$

That is, by transforming a given model to a very simple (uncorrelated) counterpart, the complex task of estimation is greatly simplified.

The problem, of course, is that solving the eigendecomposition to determine this ideal change of basis is extremely difficult, normally much *more* difficult than solving for the estimates $\hat{\underline{z}}$ themselves.

The idea, rather, is to select a *suboptimal* change of basis, without solving an eigendecomposition, examples of which are discussed in later parts of this chapter.

Implicit Basis Change:

In those cases where a suitable orthogonal transformation cannot be found, or where the measurement model is sparse or otherwise irregular, the explicit transformation of (8.2) may not be appropriate. It is possible, however, to implicitly reformulate an estimation problem in a transformed domain without the availability of an invertible transformation. Consider, for example, a standard estimation problem, recast as a linear system:

$$\begin{aligned} \hat{\underline{z}} = \left(P^{-1} + C^T R^{-1} C\right)^{-1} C^T R^{-1} \underline{m} \quad &\Rightarrow \quad \left(P^{-1} + C^T R^{-1} C\right) \hat{\underline{z}} = C^T R^{-1} \underline{m} \\ &\Rightarrow \quad A\hat{\underline{z}} = \underline{b} \end{aligned} \tag{8.6}$$

The ease with which this linear system can be solved[2] is a function of the distribution of the eigenvalues of A, known as its *spectrum*. One quantification of eigenvalue spread is via the condition number $\kappa(A)$, which follows from our earlier discussion of such inverse problems in Section 2.3.

If we now suppose a change of basis $\underline{z} = S\underline{\bar{z}}$, then we obtain a modified linear system, analogous to (8.2):

$$\begin{array}{ccccc} A\underline{z} = \underline{b} & \longrightarrow & S^T A\, S\underline{\bar{z}} = S^T\underline{b} & \longrightarrow & \bar{A}\underline{\bar{z}} = \underline{\bar{b}} \\ & & & & \downarrow \text{ Easy ?} \\ \underline{\hat{z}} & \longleftarrow & \text{S} & & \underline{\hat{\bar{z}}} \end{array} \tag{8.7}$$

The key question, then, is how the condition number $\kappa(\bar{A})$ of the modified system compares with $\kappa(A)$ of the original. To be sure, an interest in solving linear systems goes far beyond statistical image processing; indeed, this process of basis change for the purpose of improving conditioning has a significant history in the algebra literature and is known as *preconditioning* [39, 79, 98, 347].

There are several important aspects to observe regarding this preconditioning approach, in contrast to the explicit change of (8.2):

1. The change of basis occurs only in the space of the unknowns $\underline{z}$, *not* in the measurement space $\underline{m}$. Thus no assumptions are made regarding the spatial distribution or sparsity of the measurements.

2. The preconditioned approach (8.7) never requires the evaluation of the forward transform S^{-1}. All that is required is a transformation S and its transpose S^T, normally expressed implicitly as an algorithm rather than as a matrix, .

3. It is not required that S be invertible; indeed, S does not even have to be square. However, for $\bar{A} = S^T A\, S$ to be well conditioned $\bar{A}$ must be invertible, in which case S must have full column rank. Therefore S is either square-invertible, for a *change* of basis, or rectangular-full column rank, for a *reduction* of basis.

4. The implicit formulation of (8.7) applies only to estimation, and does not extend to other aspects of statistical inference and sampling.

[2] Methods discussed in Chapter 9.

8.2 Reduction of Basis

It is not obvious that the number of unknowns in the transformed space should equal the number in the original space. In particular, if $\underline{z}$ is known to be very smooth, then arguably relatively few coefficients might be sufficient to represent $\underline{z}$ very well, in contrast to a case involving a highly irregular or rough field.

In the implicit case (8.7) no assumption was made regarding the squareness or invertibility of S, and the use of a reduced basis is straightforward. Of greater interest in reduced-order modelling is the explicit case.

We start by considering the degree of approximation introduced by a reduction of basis. Suppose we propose transformations

$$\underline{\bar{z}} = F\underline{z} \qquad \underline{z} = S\underline{\bar{z}}, \tag{8.8}$$

where F, S are rectangular matrices applied to the random field $\underline{z} \sim P$, such that the size of $\underline{\bar{z}}$ is deliberately reduced. Then the mean-square error in reduced representation is

$$\begin{aligned} \mathrm{MSE}(\underline{z} - SF\underline{z}) &= E\left[(\underline{z} - SF\underline{z})^T(\underline{z} - SF\underline{z})\right] && (8.9)\\ &= E\left[\mathrm{tr}\left((I - SF)^T\underline{z}^T\underline{z}(I - SF)\right)\right] && (8.10)\\ &= \mathrm{tr}\left((I - SF)^T(I - SF)P\right). && (8.11) \end{aligned}$$

Thus under a regular change of basis (8.2), where $S = F^{-1}$, then $(I - SF) = \mathbf{0}$ and the error in representation is zero, as expected. With a reduction of basis, F^{-1} does not exist, in which case $(I - SF) \neq \mathbf{0}$, normally[3] implying that the reduction of basis introduces some degree of approximation.

Next, it is desired that the original space be able to represent perfectly any signal from the reduced space. That is, repeated projections of the form

$$\text{reduced} \;\Rightarrow\; \text{original} \;\Rightarrow\; \text{reduced}$$

should not incur additional error:

$$\begin{aligned} 0 = \mathrm{MSE}(\underline{\bar{z}} - FS\underline{\bar{z}}) &= E\left[(\underline{\bar{z}} - FS\underline{\bar{z}})^T(\underline{\bar{z}} - FS\underline{\bar{z}})\right] && (8.12)\\ &= E\left[\mathrm{tr}\left((I - FS)^T\underline{z}^T\underline{z}(I - FS)\right)\right] && (8.13)\\ &= \mathrm{tr}\left((I - FS)^T(I - FS)\bar{P}\right) && (8.14) \end{aligned}$$

from which it follows that $FS = I$, for which a sufficient condition is that F and S form what is known as a *pseudoinverse* pair.[4]

[3] If the covariance P is singular then it is possible to design a basis reduction that is exact, with no approximation.

[4] More specifically, in this case F is referred to as a *left-inverse* of S; the pseudoinverse condition, discussed in Appendix A.9, is slightly more general.

Next, we have seen (8.5) the convenience associated with orthogonal transformations. Suppose that F and S are rectangular portions of orthogonal matrices $\bar{F}, \bar{S}$:

$$\bar{F} = \begin{bmatrix} F \\ F_o \end{bmatrix} \qquad \bar{S} = \begin{bmatrix} S & S_o \end{bmatrix}, \tag{8.15}$$

such that

$$\underbrace{\bar{F} = \bar{S}^{-1} = \bar{S}^T}_{\text{Orthogonality}} \qquad \text{and} \qquad \underbrace{FS = I}_{\text{Pseudoinverse}}. \tag{8.16}$$

Then the mean-square error in the reduced representation is

$$\begin{aligned} \text{MSE}(\underline{z} - SF\underline{z}) &= \text{MSE}(\bar{S}\bar{F}\underline{z} - SF\underline{z}) = \text{MSE}(S_o F_o \underline{z}) && (8.17)\\ &= E\left[\underline{z}^T F_o^T S_o^T S_o F_o \underline{z}\right] && (8.18)\\ &= \text{tr}\left(S_o F_o P F_o^T S_o^T\right) && (8.19)\\ &= \text{tr}\left(S_o^T S_o F_o P F_o^T\right) && (8.20)\\ &= \text{tr}\left(F_o P F_o^T\right) && (8.21)\\ &= \text{MSE}(F_o \underline{z}) && (8.22) \end{aligned}$$

which is essentially a restatement of Parseval's theorem [243]: because of the orthogonality of the transformation, the degree of error in approximating $\underline{z}$ equals the mean-square error of the *omitted* basis elements $F_o\underline{z}$. Thus a "good" basis reduction is one in which the variance of $F_o\underline{z}$ is minimized.

The following sections begin with the optimum reduction of basis, based on an eigendecomposition, which is useful in dimensionality reduction, but inapplicable more generally for reasons of computational complexity. The two remaining sections will discuss suboptimal approaches to basis reduction.

Section	Method	Basic Assumption
8.2.1	Principal Components	Problem stationarity along some dimension
8.2.2	Fast Pseudoinverses	Spatial smoothness
8.2.3	Local Processing	Local correlations or dense measurements

8.2.1 Principal Components

The method of *Principal Components* [91, 165, 166, 179, 250] is one of the most fundamental in statistical analysis, variously known as the Karhunen–Loève transform, the Hotelling transform, or empirical orthogonal functions.

Given a random vector $\underline{z} \sim P$, we seek an efficient reduced-order representation $\bar{\underline{z}} = F\underline{z}$, essentially a compression of $\underline{z}$. There are two basic perspectives on the formulation of a criterion for F:

1. Find the linear transformation which captures the most statistical variability. For a single linear function $\bar{z} = \underline{f}^T \underline{z}$, the first principal component of $\underline{z}$ is the choice of $\underline{f}$ which maximizes the variance of $\bar{z}$:

$$\text{Find } \underline{f} \text{ to maximize } J(\underline{f}) = \text{var}(\bar{z}) = \underline{f}^T P \underline{f} \quad \text{such that } \underline{f}^T \underline{f} = 1 \tag{8.23}$$

   This generalizes to finding the first n principal components as the choice of transformation F which maximizes the overall variability of $\underline{\bar{z}}$, measured as the determinant of its covariance:

$$\text{Find } F \text{ to maximize } J(F) = \det\big(\text{cov}(\underline{\bar{z}})\big) = |FPF^T| \quad \text{such that } |FF^T| = 1 \tag{8.24}$$

2. Find the linear transformation which minimizes the mean-squared error in the reduced-order representation of $\underline{z}$.

   Given the reducing transformation $\underline{\bar{z}} = F\underline{z}$, the estimator $\underline{\hat{z}}$ which minimizes the mean-squared error in $(\underline{\hat{z}} - \underline{z})$ is

$$\underline{\hat{z}} = F^T (FF^T)^{-1} \underline{\bar{z}}, \tag{8.25}$$

   clearly a function of F. The optimal choice of F is the one minimizing this MSE:

$$\text{Find } F \text{ to minimize } J(F) = \text{MSE}(\underline{\hat{z}} - \underline{z}). \tag{8.26}$$

Both of the above criteria lead to the same conclusion:

> *The principal components are given by the eigenvectors of P. That is, the optimum linear reducing transformation is found by letting the rows of F equal the eigenvectors corresponding to the q largest eigenvalues of P.*

Given the eigendecomposition

$$P\underline{v}_i = \lambda_i \underline{v}_i \qquad \lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n \geq 0 \tag{8.27}$$

where we define the reduction matrix

$$V_q = [\underline{v}_1 \ldots \underline{v}_q], \tag{8.28}$$

then the principal-component representation of $\underline{z}$ is

$$\underline{\bar{z}} = V_q^T \underline{z}, \tag{8.29}$$

where the mean-square representation error in keeping these first q principal components is given by the sum of the omitted eigenvalues

$$\text{MSE}(\underline{z} - SF\underline{z}) = \text{MSE}(\underline{z} - V_q V_q^T \underline{z}) = \sum_{j=q+1}^{n} \lambda_j. \tag{8.30}$$

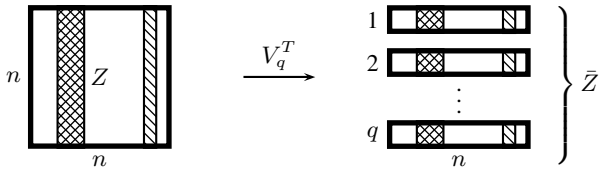


Fig. 8.3. Dimensionality reduction of stationary fields: a two-dimensional image Z is transformed into q decoupled problems $\bar{Z}$.

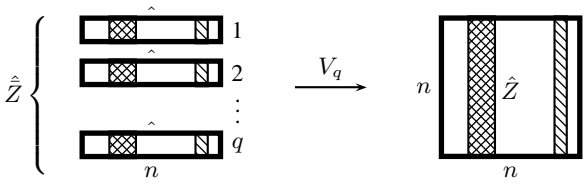


Fig. 8.4. Dimensionality restoration of an estimated reduced-order model: after estimates have been computed for the q decoupled problems in $\bar{Z}$ the estimates in the original domain Z are easily found, inverting the procedure of Figure 8.3.

If we consider $\underline{z}$ to be a lexicographically-stacked two- or higher-dimensional random field, then the size of $\underline{z}$ precludes applying an eigendecomposition, which has complexity $\mathcal{O}(n^3)$, directly. However principal components can be applied to a single dimension in a multidimensional problem. In particular, suppose that $Z = \left[\underline{z}_1\, \underline{z}_2 \ldots \underline{z}_n\right]$ is a two-dimensional $n \times n$ random field, as shown in Figure 8.3. For dimensionality reduction to be applicable, Z and its associated inverse problem must satisfy two assumptions:

1. The columns of Z must be statistically stationary, such that $P = \mathrm{cov}(\underline{z}_i)$ is not a function of i, meaning that the principal components will not vary from column to column.
2. A given column of Z must be either densely measured, or not at all, to allow an explicit change of basis.

Let V_q be the qth-order reduction matrix, the q most significant eigenvectors of P, then

$$\bar{Z} = V_q^T Z \tag{8.31}$$

is the reduced $n \times q$ two-dimensional random field, with the key property that the rows are decorrelated and can be processed separately. Because of the orthogonality of the eigendecomposition the inversion of the reduction step is very easy:

$$\hat{Z} = V_q \hat{\bar{Z}}, \tag{8.32}$$

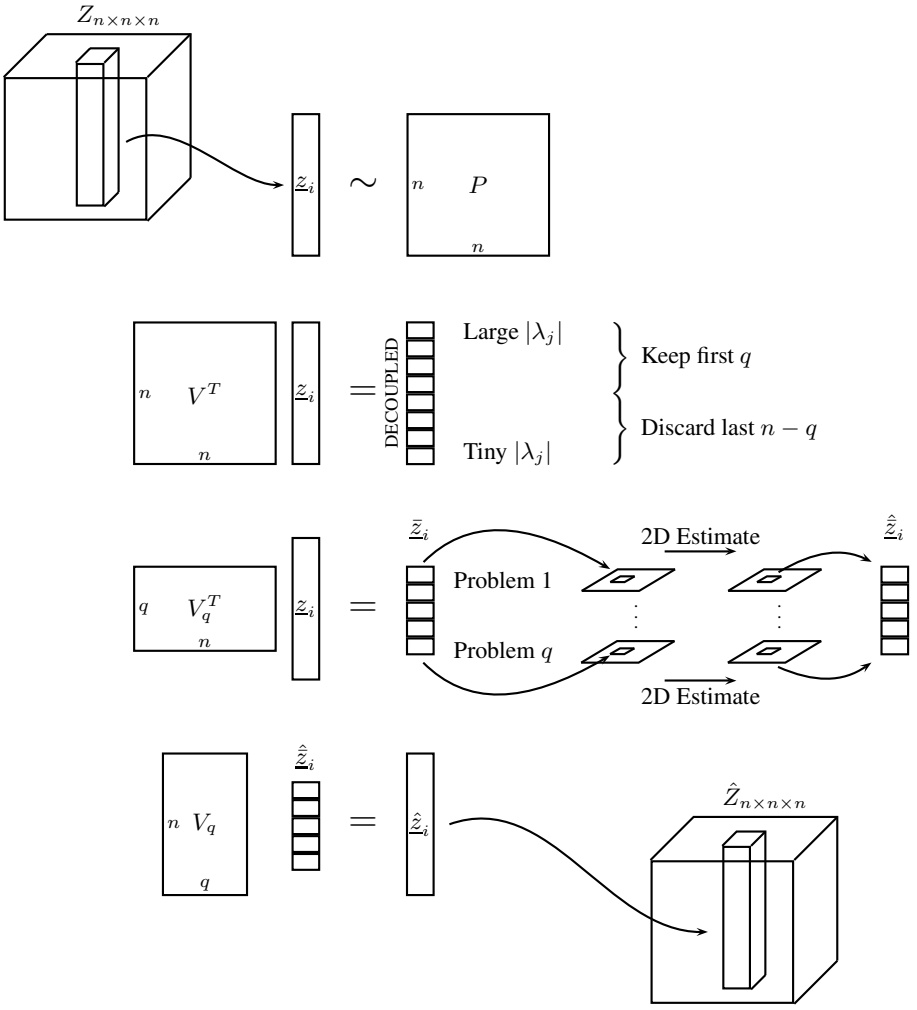
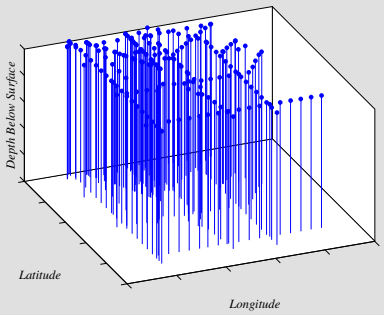


Fig. 8.5. Dimensionality reduction in 3D: If the columns of a volume Z are statistically stationary, we can use principal components to divide the volume into q decoupled 2D slices, each of which is solved separately, and then recombined to form the estimated volume $\hat{Z}$.

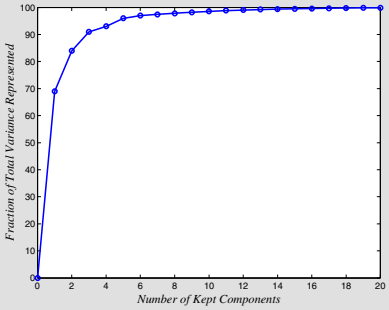
shown in Figure 8.4. Obviously the above development generalizes to reducing a d-dimensional problem to a set of q separate $(d-1)$-dimensional problems, as is illustrated for three dimensions in Figure 8.5.

Example 8.1: Principal Components for Remote Sensing

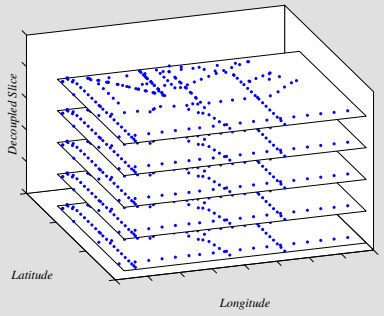
We consider an example in three-dimensional remote sensing [232]. Suppose we have measurements of ocean temperature, sparsely sampled in location, but uniformly sampled in depth:



The ocean statistics are highly variable in depth, but approximately stationary in space, so we can consider taking principal components along the depth axis. To determine the number of components needed we examine the summed eigenvalues:



Four to ten components are enough to keep 95% to 98% of the signal energy; we'll keep five, which leaves us with five sparse, decoupled two-dimensional problems to solve:



We will see methods for solving this estimation problem in Chapters 9 and 10, with the resulting estimates plotted in Figure 10.10 on page 346.

The application of principal components is made tractable here in that it is applied to only a single column of state elements, not over the entire domain. The size and complexity of the problem is reduced as follows:

	Original Space			Reduced Space		
	# Problems	Size	Complexity	# Problems	Size	Complexity
2D	1	$n \times n$	$\mathcal{O}(n^6)$	q	$1 \times n$	$\mathcal{O}(qn^3)$
3D	1	$n \times n \times n$	$\mathcal{O}(n^9)$	q	$n \times n$	$\mathcal{O}(qn^6)$

Although each of the q decoupled problems is still a spatial statistical problem, the relationship of the $(d-1)$-dimensional spatial models in $\bar{Z}$ to the d-dimensional model in Z is generally unclear. Even if Z were stationary in all dimensions, the q transformed problems would not be governed by a single model: a separate model needs to be derived for each of the q new decoupled problems. In general, rather than attempting to transform a model from Z to $\bar{Z}$, it is normally simpler to learn the statistical models in $\bar{Z}$ directly, for example from simulated or measured data.

8.2.2 Multidimensional Basis Reduction

The previous section outlined the use of principal components in cases where a multidimensional problem is stationary, allowing it to be decoupled into a number of uncorrelated, smaller pieces.

In those cases where the model is spatially nonstationary, or where a decoupling into separate problems is inconvenient or undesirable, one alternative is not to divide the problem into multiple pieces, rather to formulate a reduced-order version of the full problem. That is, we wish to consider basis reduction: representing a large problem $\underline{z} \in \mathbb{R}^n$ in a much smaller, reduced domain $\underline{\bar{z}} \in \mathbb{R}^q$.

As before, we consider a transformation between an original random field $\underline{z}$ and its reduced counterpart $\underline{\bar{z}}$,

$$\underline{\bar{z}} = F\underline{z} \qquad \underline{z} = S\underline{\bar{z}}, \tag{8.33}$$

where F, S are subsampling/reduction and interpolating transforms, respectively. In principle, F and S may be space-variant, irregular operators. In practice such generality is unneeded and overcomplicated for large random fields; instead, it is simpler to think of $\underline{\bar{z}}$ as representing a low- or coarse-resolution version of $\underline{z}$, such that $\underline{\bar{z}}$ lives on a regular grid or lattice, organized in the same way as the lattice for $\underline{z}$, but downsampled by a factor γ in each dimension. For large random fields we are unlikely to specify F or S directly; instead, it is simpler to suppose that the operations are generated by space-invariant kernels $\mathcal{F}, \mathcal{S}$, such that each element in $\underline{\bar{z}}$ essentially introduces a weighted copy of $\mathcal{S}$ in the corresponding parts of $\underline{z}$. We can write this as

$$\bar{Z} = \downarrow(\mathcal{F} * Z) \qquad Z = \mathcal{S} * \left(\uparrow \bar{Z}\right), \tag{8.34}$$

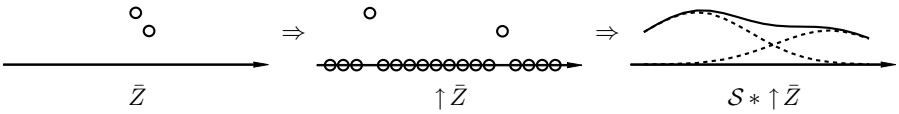


Fig. 8.6. The transformation from a coarse representation, left, via zero-padding, middle, and convolution, right, with a Gaussian kernel $\mathcal{S}$.

where $\downarrow$ and $\uparrow$ represent subsampling and zero-padding, respectively. The interpolating kernel $\mathcal{S}$, illustrated in Figure 8.6, is the more intuitive of the two. Figure 8.7 illustrates a two-dimensional example, in which a smooth random field is represented by a low-resolution "image", a $\gamma = 6$ reduction to a 5×5 grid of extracted coefficients.

A few considerations for the boundaries of the domain of Z:

1. Although many mathematical functions have infinitely long tails, such as Gaussians, in practice the support of $\mathcal{S}$ must be of finite size, so long kernels must be truncated.

2. It is often efficient to use the FFT (the fast Fourier transform) to compute convolutions, such as in (8.34). As the FFT actually implements a *circular* convolution, additional zero-padding may be needed to prevent a wrapping around the ends of the lattice, where the extent of zero-padding is proportional to the size of support of $\mathcal{S}$.

3. In most cases, the interpolating kernel needs to be different near boundaries than inside the domain. That is, the interpolator is not, in fact, stationary. However, rather than abandoning the kernel concept and specifying S explicitly, it is simpler to specify a nonstationary modification of the stationary convolution:

$$Z = \big(\mathcal{S} * \big(\uparrow \bar{Z}\big)\big) \oslash \big(\mathcal{S} * (\uparrow \mathbf{1})\big). \tag{8.35}$$

   Specifically, this ensures that a reduced field of all ones interpolates to a fine-scale field of all ones. Many other normalizations are possible.

So what sort of kernel $\mathcal{F}, \mathcal{S}$ or associated induced transformation F, S makes an effective change of basis?

Clearly an appropriate choice of basis will have to be a function of the statistics of the random field; although specialized approaches can be developed for particular statistics, here we limit our attention to the reduction of *smooth* random fields. Thus the interpolating basis elements, the kernel $\mathcal{S}$ or the columns of S, should be spatially smooth. There are three criteria to consider:

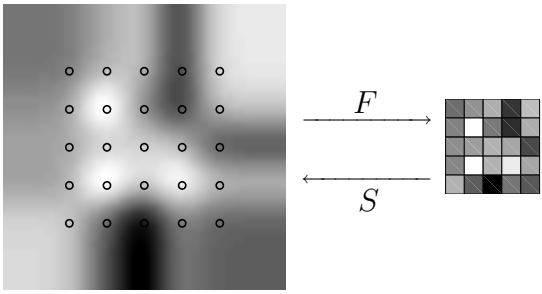


Fig. 8.7. We can represent a large random field, left, using a reduced set of coarse-scale coefficients, right. A Gaussian shape was used here as the interpolating kernel $\mathcal{S}$.

1. PSEUDOINVERSE: Choosing F and S to be a pseudoinverse pair (Appendix A.9) provides two benefits:

   - Choosing $F = S^+$ minimizes the representation mean-squared-error,
   - The pseudoinverse criterion guarantees that $FS = I$, meaning that there is no error induced by repeated coarse–fine–coarse projections.

   The complexity of pseudoinversion makes the relationship between $\mathcal{F}$ and $\mathcal{S}$ nearly impossible to specify. The analytic form of the Moore–Penrose pseudoinverse [4] is well known, but computationally intractable for large problems:

$$S = F^+ = F^T(FF^T)^{-1} \qquad F = S^+ = (S^TS)^{-1}S^T. \tag{8.36}$$

   Instead, the pseudoinverse should be represented *implicitly*. Either F or S may be specified, and the other inferred. For example, if we assume the interpolator S to be a given huge, sparse matrix, then we compute

$$Q = S^TS, \tag{8.37}$$

   where the matrix multiplication S^TS is straightforward because of the spatial sparsity of S, and where the size of Q is the size of the reduced state. Then the pseudoinverse is computed as

$$\bar{\underline{z}} = F\underline{z} = S^+\underline{z} = Q^{-1}(S^T\underline{z}) = Q^{-1}\underline{z}_s \tag{8.38}$$

   such that the matrix product $Q^{-1}S^T$ is never explicitly calculated.

   In the event that Q is too large to invert, or Q^{-1} too large to store, $\bar{\underline{z}}$ can be solved iteratively [111] from the linear system

$$Q\bar{\underline{z}} = \underline{z}_s. \tag{8.39}$$

2. NOISE INSENSITIVITY: A second desired property is that the transformations be stable with respect to errors. Consider, for example, the degree to which a perturbation δ at the fine scale affects the coarse-scale coefficients:

$$\underline{\bar{z}} \xrightarrow{S} (\underline{z} + \delta) \xrightarrow{F} \underline{\bar{z}}_\delta \tag{8.40}$$

If S has a tiny singular value, then its pseudoinverse F must have a corresponding large singular value, implying that a small disturbance δ could be amplified by F to give rise to arbitrarily large differences in $(\underline{\bar{z}} - \underline{\bar{z}}_\delta)$, leading to a normalized noise sensitivity criterion

$$\frac{|\underline{\bar{z}} - \underline{\bar{z}}_\delta|}{|\delta|} \frac{|\underline{z}|}{|\underline{\bar{z}}|} \equiv \frac{|F\delta|}{|\delta|} \frac{|S\underline{\bar{z}}|}{|\underline{\bar{z}}|}. \tag{8.41}$$

The upper bound for this sensitivity is given by the product of the largest singular values $\sigma_{\max}$ of F and S,

$$\sigma_{\max}(F) * \sigma_{\max}(S) = \text{cond}(F) = \text{cond}(S). \tag{8.42}$$

That is, the noise sensitivity is bounded by the condition number of the subsampler F, equivalently that of the interpolator S, implying that the noise sensitivity can be evaluated from either of F, S without computing a pseudoinverse.

In general, an interpolating kernel $\mathcal{S}$ that passes through zero may not at all (or just barely) sample certain fine-scale elements, making the problem nearly singular. A kernel robust to noise sensitivity should therefore be strictly positive, or must be used in a subsampling geometry which guarantees that kernel zeros of neighbouring coarse-scale elements do not coincide.

3. SHIFT INVARIANCE: A third desired attribute is that the basis-reduction operation be shift-invariant, meaning that the particular choice of origin for the coarse scale (that is, how the coarse lattice is arranged relative to the fine, as in Figure 8.7) should not lead to spatial variations in the quality or type of representation. In other words, if I can represent a fine-scale field $Z(\underline{x})$, I should equally well be able to represent its shifted version $Z(\underline{x} - \underline{\delta})$, meaning that translation and representation commute:

$$T\,SF\underline{z} \approx SF\,T\underline{z}, \tag{8.43}$$

where T is a spatial translation on the fine scale. There are two motivations for this assertion:

   1. It is undesirable for the coarse grid to manifest itself in any explicit way in the inversion $S\underline{\bar{z}}$.

   2. In many cases the random field is actually time-dynamic $\underline{z}(t)$, but where the dynamics may be slow, involving shifts and motions much smaller than the coarse discretization interval γ. The sampling-interpolation operation SF should therefore be insensitive to spatial shifts to ensure that a slow, advective flow is not progressively corrupted by repeated sampling and interpolation.

Shift insensitivity, (8.43), implies that for any coarse field $\underline{\bar{z}}$ and fine-scale shift operation T, there exists a new coarse field $\underline{\bar{z}}_T$ corresponding to the shifted field:

$$TS\underline{\bar{z}} = S\underline{\bar{z}}_T \tag{8.44}$$

and thus, ignoring boundary effects,

$$T\big(\mathcal{S} * (\uparrow \underline{\bar{z}})\big) = \big(\mathcal{S} * (\uparrow \underline{\bar{z}}_T)\big). \tag{8.45}$$

As convolution simplifies to multiplication in the frequency domain, taking the Fourier transform of (8.45) leads to

$$F(\mathcal{S})F(\uparrow \underline{\bar{z}})F(T) = F(\mathcal{S})F(\uparrow \underline{\bar{z}}_T). \tag{8.46}$$

For this to be satisfiable for all $\underline{\bar{z}}$, $F(\mathcal{S})$ must be mostly zero: essentially, the interpolating kernel $\mathcal{S}$ must be bandlimited, or smooth.

It is important to note that, in general, these latter two criteria are in opposition. That is, a highly band-limited kernel is typically poorly conditioned, and thus sensitive to noise, whereas a kernel corresponding to a low condition number is typically poorly bandlimited, and thus exhibits shift sensitivities.

There is one significant exception to this opposition: if the domain has periodic boundary conditions and is stationary, then the Fourier transform offers a perfect pseudoinverse pair

$$\underline{\bar{z}} = \mathrm{FFT}^{-1}\Big(\mathrm{trunc}\big(\mathrm{FFT}(\underline{z})\big)\Big) \tag{8.47}$$

$$\underline{z} = \mathrm{FFT}^{-1}\Big(\mathrm{zeropad}\big(\mathrm{FFT}(\underline{\bar{z}})\big)\Big), \tag{8.48}$$

where truncation and zero-padding are frequency domain operations implementing an ideal low-pass filter. In general the rigid assumptions (stationarity and boundary periodicity) limit the usefulness of this approach, although we encounter the FFT again in Section 8.3.

We conclude this section with a brief survey of possible interpolating kernels with respect to the condition-number and bandlimit criteria. At first glance the kernels of Figure 8.8 may appear to be superficially similar to the analytical statistical kernels of Figure 5.16 on page 161; it is crucial to understand a fundamental difference: the kernels $\mathcal{P}$ of Figure 5.16 represent a covariance, and therefore *must* satisfy positive-definiteness, whereas the kernels $\mathcal{S}$ of Figure 8.8 represent a transformation, which we would like to have well-conditioned and bandlimited, but in fact *any* transformation is permissible. That is, one can arbitrarily experiment with different choices of $\mathcal{S}$, whereas most choices of $\mathcal{P}$ will fail to be positive definite.

As a qualitative evaluation, Table 8.1 plots the shift sensitivity and condition number for the eight kernels of Figure 8.8. It must be understood that these numbers are illustrative only, as they can vary greatly with problem size and geometry. Two observations:

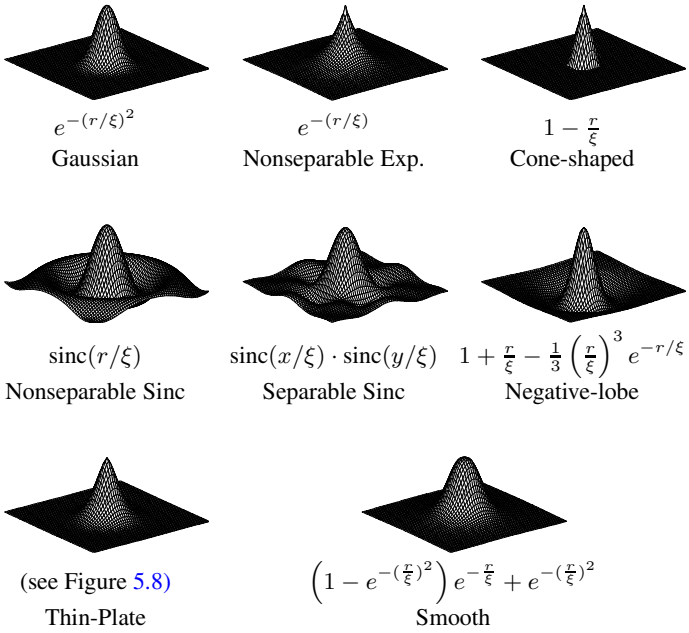


Fig. 8.8. Eight plausible interpolation kernels $\mathcal{S}$; in all cases $r = \sqrt{x^2 + y^2}$ measures the distance to the origin and ξ is a scale parameter that controls the spatial size of the interpolator.

1. The kernel of a perfect low-pass filter is the separable-sinc function, so it is interesting to see the significant difference in behaviour between it and the FFT. The differences stem from the boundary periodicity of the FFT and the very slow decay of the sinc kernels: the finite domain tested in Table 8.1 has non-periodic boundaries, and for computational reasons the kernels are truncated to a finite size, particularly problematic for slow-decay kernels.

2. The Gaussian kernel is unique, maximally bandlimited simultaneously in space and frequency, thus well-approximated as a truncated kernel (space bandlimit) and giving excellent shift-sensitivity (frequency bandlimit).

A remote-sensing application of multidimensional basis reduction is shown at the end of this chapter on page 285.

Kernel $\mathcal{S}$	log Shift Sensitivity	log Condition Number
Gaussian	$0-2$	$2-4$
Smooth	3	$1-2$
Thin-Plate	3.3	1
Nonseparable Exponential	3.5	1
Negative lobe	3.5	$1-2.5$
Separable Sinc	3.5	1
Nonseparable Sinc	3.8	$1-2$
Cone-shaped	3.8	1
FFT (periodic separable sinc)	$-\infty$	0

Table 8.1. Shift sensitivity and condition number, evaluated numerically for the eight kernels of Figure 8.8 on a 10×10 two-dimensional domain.

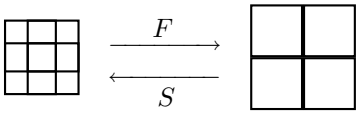
8.2.3 Local Processing

A final choice of basis reduction is the relatively simple approach of local processing, in which only a small portion of the overall problem is solved at a time, and the results stitched together. Clearly the computational savings can be substantial; for an algorithm with cubic complexity, dividing a problem into q equally sized pieces, each of size $1/q$ and complexity $1/q^3$, results in a computational complexity $1/q^2$ as great as solving the full problem directly.

Obviously this sort of approach admits many variations, a few of which are illustrated in Figure 8.9. These approaches are, in general, most applicable to estimation, and less so to sampling, because measurements will tend to cause the estimates in adjacent blocks to be similar, whereas there is nothing to force any consistency between blocks in the stochastic (random) variations in sampling.

The overlapped approach [169] is a significant special case: because the estimate of a single state element z_i is best estimated based on measurements *around* i, and not just *at* i, it is preferable to divide the problem into overlapping blocks, allowing adjacent blocks to be interpolated, limiting inter-block artifacts and discontinuities.

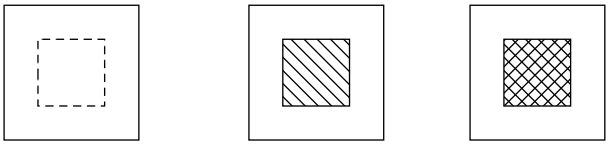
The overlapped approach is, itself, essentially a projection onto a new basis, this time a projection into a redundant domain, as some state elements will belong to multiple overlapped blocks:



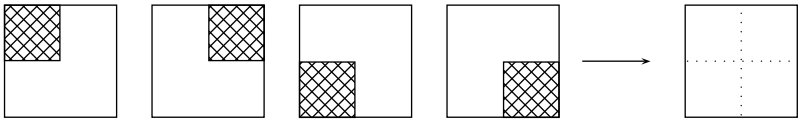
(8.49)

such that, as usual, $SF = I$. The size, number, and degree of overlap of the blocks uniquely define F:

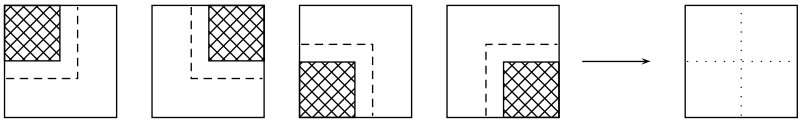
Consider solving an estimation problem by processing local regions separately. We illustrate the process graphically using the following notation:



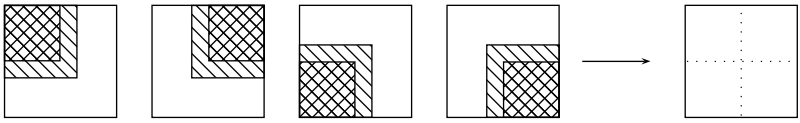
First we can divide the problem into disjoint pieces, stitching together the individual estimates to obtain the overall result:



The stitched results will tend to be blocky, because there is nothing to force continuity at the region boundaries. Because a pixel on the edge of a region would benefit from nearby measurements outside of the region, we can extend the range of measurements used:



In some cases it is inconvenient to use measurements at locations which are not estimated. We could, instead, estimate over a larger region, but only keep the more local estimates:



Having generated estimates in overlapping regions, why not take advantage of them? We can synthesize the final result as a tapered interpolation from one region to the next:

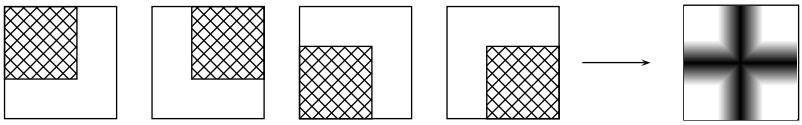


Fig. 8.9. Basis reduction by local processing: We can solve a larger problem by dividing it into separated subproblems. The overlapped approach, bottom, performs no spatial blurring; any interpolation is between multiple estimates at a single location.

$$F_{i,j} = \begin{cases} 1 & \text{if } \underline{\bar{z}}_i \text{ corresponds to element } \underline{z}_j \\ 0 & \text{otherwise} \end{cases} \tag{8.50}$$

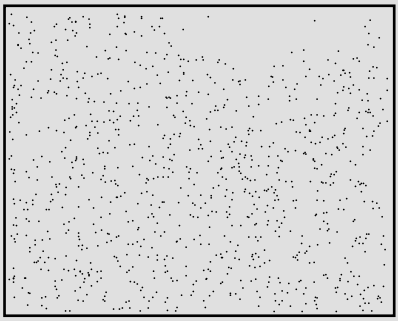
S is not uniquely defined, as there are many possible ways to interpolate the redundant state elements. A spatial linear interpolation from one block to the next [169] is a simple, intuitive choice. The resulting estimates are computed as

$$\underline{z} \xrightarrow{F} \underline{\bar{z}} \xrightarrow{\text{Divide}} \{\underline{\bar{z}}_i\} \xrightarrow[\text{Estimation}]{\text{Local}} \{\underline{\hat{\bar{z}}}_i\} \xrightarrow{\text{Combine}} \underline{\hat{\bar{z}}} \xrightarrow{S} \underline{\hat{z}}. \tag{8.51}$$

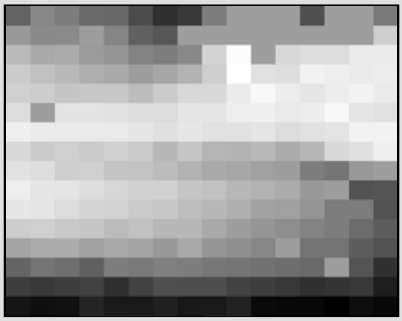
It is important to understand that operators F and S do *not* blur or smooth; they are pointwise operators. The overlapped results in Example 8.2 give the appearance of having been blurred, however the smoothness is due to the spatial constraints on $\underline{\bar{z}}$ in the overlapped domain; there is no spatial averaging.

Example 8.2: Overlapped Local Processing

We wish to consider local estimation, as in Figure 8.9. Suppose we have sparse satellite temperature measurements of the eastern equatorial Pacific ocean. We will propose a very simple local estimator: we set the estimate of all pixels in a block equal to the mean of the measurements which lie within it.



Sparse Data Points



Disjoint-Block Estimates

As was discussed in Section 8.2.3, and is very clear here, local estimation based on disjoint blocks suffers from inter-block artifacts (discontinuities). To address this we can define overlapped blocks, such that each block is processed independently, but the resulting estimated image is found as a tapered interpolation. The block size and inter-block overlap must satisfy

$$\text{Image Size} = \#\,\text{Blocks} \cdot \text{Block Size} - (\#\,\text{Blocks} - 1) \cdot \text{Overlap}. \tag{8.52}$$

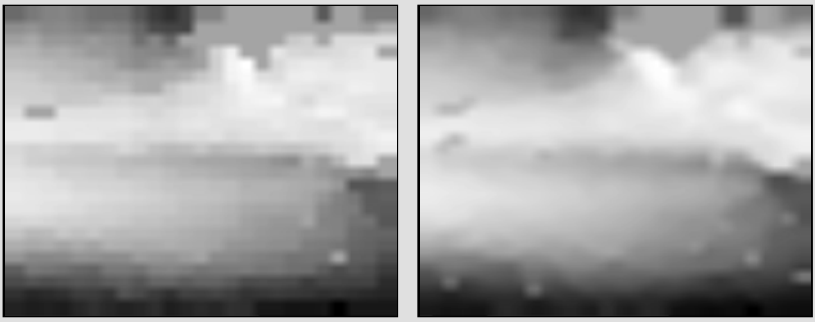
Example continues ...

Example 8.2: Overlapped Local Processing (cont'd)

As all of these quantities are integer only certain values are permitted, most easily found by testing (8.52) numerically for a wide range of integer possibilities. For the 512×512 image being processed here we select two cases:

Image Size	# Blocks	Block Size	Overlap (pix)	Overlap (%)
512	25	32	8	37%
512	41	32	20	63%

leading to the following results:



37% Overlap 63% Overlap

8.3 FFT Methods

A very special change of basis is the Fourier basis for stationary, periodic random fields [83, 276]. In particular, the Fourier basis elements are the eigenvectors, and thus the perfect change of basis, for *every* stationary, periodic field. Although such stationary, periodic fields may be rare in practice, the methods available to solve them are so efficient and elegant that some discussion is merited.

A finite, one-dimensional stationary random process with periodic boundary conditions is known as *circulant*, meaning that the points of the process can be visualized as lying on a circle: no beginning, no end. Similarly in two dimensions, a stationary random field on a rectangular lattice with periodic boundary conditions is known as *toroidal*,[5] as implied in Figure 8.10.

The correlation structure of a d-dimensional stationary, periodic $n_1 \times \cdots \times n_d$ random field therefore takes the form

$$E\left[z_{\underline{i}} z_{(\underline{i}+\underline{\delta}) \bmod \underline{n}}\right] = E\left[z_{0,0} z_{\underline{\delta} \bmod \underline{n}}\right] = \mathcal{P}_{\underline{\delta}}, \tag{8.53}$$

where

[5] That is, topologically, the wrapping of a rectangular sheet onto a torus or doughnut.

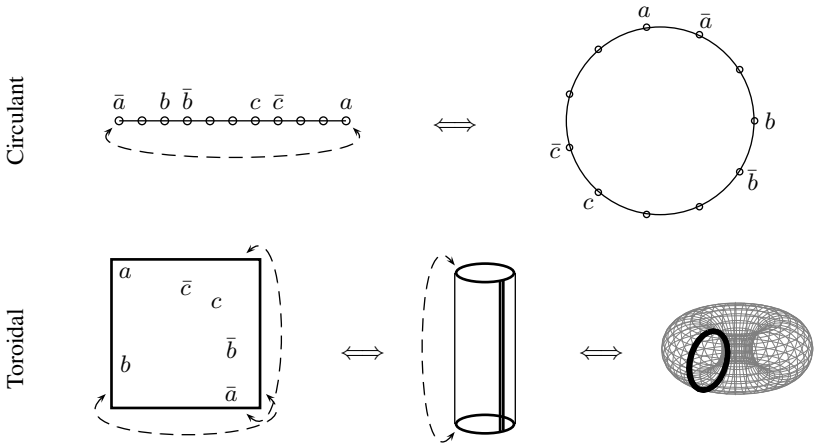


Fig. 8.10. Illustration of periodic, stationary random processes in one and two dimensions. In one dimension the problem maps to a circle, and in two dimensions to a torus, such that the point pairs $(a\ \bar{a}), (b\ \bar{b}), (c\ \bar{c})$ all have the same joint statistics.

$$\underline{i} \bmod \underline{n} = \left[(i_1 \bmod n_1), \ldots, (i_d \bmod n_d)\right]^T. \tag{8.54}$$

If the process is lexicographically ordered, the covariance of the resulting random vector will have a special circulant or block-circulant structure [83], as illustrated in Figure 8.11. The special significance of any such process is that its covariance is diagonalized by the d-dimensional FFT.

8.3.1 FFT Diagonalization

Define the Fourier basis element of length N as

$$\underline{f}_N^T = \left[e^{-j2\pi 0/N} \ \cdots \ e^{-j2\pi(N-1)/N}\right]. \tag{8.55}$$

We first consider the diagonalization of a one-dimensional random process $\underline{z}$ with circulant covariance

$$\underline{z} \sim P = \left[\underline{p}_0 \ \cdots \ \underline{p}_{N-1}\right]. \tag{8.56}$$

The Fourier transform, or FFT, applied to $\underline{p}_0$ is

$$\mathrm{FFT}(\underline{p}_0) = F\underline{p}_0 \equiv \begin{bmatrix} \left(\underline{f}_N^T\right)^0 \\ \vdots \\ \left(\underline{f}_N^T\right)^{N-1} \end{bmatrix} \underline{p}_0 \equiv \bar{\underline{p}}_0. \tag{8.57}$$

$$\begin{bmatrix} a & b & c & d \\ d & a & b & c \\ c & d & a & b \\ b & c & d & a \end{bmatrix} \qquad \begin{bmatrix} A & B & C & D \\ D & A & B & C \\ C & D & A & B \\ B & C & D & A \end{bmatrix}$$

One-dimensional circulant covariance — Two-dimensional toroidal covariance

Fig. 8.11. Examples of circulant and toroidal covariance matrices [83]. In one dimension, each row or column is equal to the previous row or column, rotated by one position. In two dimensions, the lexicographical reordering of the 2D process to a one-dimensional vector implies that the covariance has a block-circulant structure (from the stationarity/periodicity of the 2D-process rows), where each block A, B, C, D is itself circulant (from the stationarity/periodicity of the 2D-process columns).

Because P is circulant, $\underline{p}_i$ is just $\underline{p}_0$ rotated downwards (Figure 8.11) by i positions, therefore the standard circular-shift property [244] of the Fourier transform applies

$$F\underline{p}_i = \bar{\underline{p}}_0 \odot \left(\left[e^{-j2\pi 0/N} \ \cdots \ e^{-j2\pi(N-1)/N} \right]^T \right)^i = \bar{\underline{p}}_0 \odot \left(\underline{f}_N^T \right)^i . \tag{8.58}$$

Therefore the Fourier transform applied to the whole covariance can be factored as

$$FP = \left[\bar{\underline{p}}_0 \odot \left(\underline{f}_N^T \right)^0 \ \cdots \ \bar{\underline{p}}_0 \odot \left(\underline{f}_N^T \right)^{(N-1)} \right] \tag{8.59}$$

$$= \left(\bar{\underline{p}}_0 \cdot \underbrace{\left[1 \ \cdots \ 1 \right]}_{N \text{ times}} \right) \odot \underbrace{\left[\underline{f}_N^0 \ \cdots \ \underline{f}_N^{N-1} \right]}_{F^T = F} . \tag{8.60}$$

Therefore the covariance of the transformed random field is

$$\begin{aligned} \underline{z} \sim P \Longrightarrow F\underline{z} \sim \bar{P} = FPF^H &= \left(\left(\bar{\underline{p}}_0 \cdot \left[1 \ \cdots \ 1 \right] \right) \odot F \right) F^H \\ &= \text{Diag}(\bar{\underline{p}}_0). \end{aligned} \tag{8.61}$$

That is, the FFT diagonalizes the covariance associated with *any* stationary, periodic random vector. Stated another way, the eigenvectors for all circulant matrices are the Fourier basis elements, and the FFT of the circulant matrix returns the eigenvalues. Thus

$$\underline{z} \sim P \text{ Circulant} \implies \text{FFT}(\underline{z}) \sim \text{Diagonal}. \tag{8.62}$$

Given a two-dimensional toroidal random field Z, each row or column of Z is circulant, so an FFT applied to the columns of Z

$$\bar{Z} = \text{FFT}(Z) = \left[\text{FFT}(\underline{z}_0) \ \cdots \ \text{FFT}(\underline{z}_{N-1}) \right] \tag{8.63}$$

has uncorrelated rows. As the FFT is a stationary, periodic operator, each row of $\bar{Z}$ is still circulant, so a second FFT applied to the rows

$$\bar{\bar{Z}} = \left(\text{FFT}(\bar{Z}^T)\right)^T \tag{8.64}$$

decorrelates the elements within rows. Therefore all elements in $\bar{\bar{Z}}$ are decorrelated from all others, meaning that the associated covariance is diagonalized. As taking an FFT by columns and then by rows is equivalent to the two-dimensional FFT, we conclude that

$$[Z]_{:} \sim P \text{ Toroidal} \implies [\text{FFT}_2(Z)]_{:} \sim \text{Diagonal}. \tag{8.65}$$

Finally, by induction we generalize to the d-dimensional case. If Z is a random field of dimensions $n_1 \times \cdots \times n_d$, then

$$Z \; d\text{-Dimensional, Stationary, Periodic} \implies [\text{FFT}_d(Z)]_{:} \sim \text{Diagonal}. \tag{8.66}$$

8.3.2 FFT and Spatial Models

It is clear that a covariance P is an inefficient representation of large, multidimensional random fields. The inefficiency is particularly striking with stationary, periodic fields, as the entire covariance $P = \left[\underline{p}_0 \, \underline{p}_1 \, \ldots\right]$ can be reconstructed from the first column $\underline{p}_0$ alone, since

$$[\mathcal{P}]_{:} = \underline{p}_0 \tag{8.67}$$

as was discussed in Section 5.3.2. We start with the covariance $\bar{P}$ in the transformed domain, from (8.61):

$$\bar{P} = \text{Diag}(\underline{\bar{p}}_0) = \text{Diag}\left(F_d \underline{p}_0\right) = \text{Diag}\left([\text{FFT}_d(\mathcal{P})]_{:}\right). \tag{8.68}$$

Now, consider finding the inverse covariance P^{-1}:

$$\bar{P} = F_d P F_d^H \implies P^{-1} = F_d^H \bar{P}^{-1} F_d. \tag{8.69}$$

The matrix inversion is easy, as $\bar{P}$ is diagonal. However, because circulant/toroidal nonsingular matrices have circulant/toroidal inverses, the explicit construction of P^{-1} makes no sense, rather we really want $\mathcal{P}^{-1}$, the kernel corresponding to P^{-1}:

$$\underline{p}_0 = F_d^H \underline{\bar{p}}_0 \implies \mathcal{P} = \text{FFT}_d^{-1}\left([\underline{\bar{p}}_0]_{\underline{n}}\right) = \text{FFT}_d^{-1}\left(\bar{\mathcal{P}}\right) \tag{8.70}$$

$$\implies \mathcal{P}^{-1} = \text{FFT}_d^{-1}\left(1 \oslash \bar{\mathcal{P}}\right) = \text{FFT}_d^{-1}\left(1 \oslash \text{FFT}_d(\mathcal{P})\right), \tag{8.71}$$

where

1. Vector $\underline{n}^T = [n_1 \, n_2 \ldots n_d]$ represents the size of the d-dimensional domain being considered,
2. Matrix inversion has been simplified to reciprocals in the transformed domain,
3. *All* operations are directly performed on kernels.

Thus we have

$$\mathcal{P}^{-1} = \text{FFT}_d^{-1}\big(1 \oslash \text{FFT}_d(\mathcal{P})\big) \qquad \mathcal{P} = \text{FFT}_d^{-1}\big(1 \oslash \text{FFT}_d(\mathcal{P}^{-1})\big). \tag{8.72}$$

If, from Chapter 5, we recognize $\mathcal{P}^{-1}$ to be a constraint or GMRF model kernel, then we have derived a significant result: for periodic random fields we have an efficient means of converting between a model and its associated correlation structure. For example, this FFT approach was used to compute the correlation length (related to $\mathcal{P}$) as a function of thin-plate model ($\mathcal{P}^{-1}$) in Figure 5.15 on page 158.

To complete the discussion, note that

$$\det(P) = \prod_i \lambda_i(P) = \prod \text{diag}(\bar{P}) = \prod \text{FFT}_d(\mathcal{P}). \tag{8.73}$$

If we are calculating log-likelihoods, for example for parameter estimation, then the log-determinant can be calculated very efficiently as

$$\log\big(\det(P)\big) = \sum \log\big(\text{FFT}_d(\mathcal{P})\big). \tag{8.74}$$

8.3.3 FFT Sampling and Estimation

Given a prior model $\mathcal{P}$ or $\mathcal{P}^{-1}$ (the distinction is unimportant, as (8.72) allows an easy conversion) we wish to generate random samples from the prior model:

$$\underline{z} = P^{1/2}\underline{w} \qquad \underline{w} \sim \mathcal{N}(\underline{0}, I) \tag{8.75}$$

$$= F^H \bar{P}^{1/2} F\underline{w} \tag{8.76}$$

$$= F^H \text{Diag}(\bar{p}_0)^{1/2} F\underline{w}. \tag{8.77}$$

Removing the lexicographic ordering on both sides of (8.77) leaves us with a simple, fast sampler:

$$Z = \text{FFT}_d^{-1}\left(\sqrt{\text{FFT}_d(\mathcal{P})} \odot \text{FFT}_d(W)\right), \tag{8.78}$$

where W is a white random field of unit-variance random values, and where W and $\mathcal{P}$ are the same size as Z.

We hope the pattern has become clear: because all circulant matrices are diagonalized in the transformed domain, operations of matrix multiplication and inversion become corresponding scalar operations on the transformed kernels. Thus the equations for least-squares estimation and posterior sampling follow by inspection. Suppose we are given a fully stationary estimation problem, meaning that the prior P, observation matrix C, and observation noise covariance R are all circulant/toroidal. Let

$$\bar{\mathcal{P}} = \text{FFT}_d(\mathcal{P}) \quad \bar{\mathcal{C}} = \text{FFT}_d(\mathcal{C}) \quad \bar{\mathcal{R}} = \text{FFT}_d(\mathcal{R}) \tag{8.79}$$

Algorithm 3 FFT Estimation and Sampling

Goals: Compute estimates, given measurements M and model $\mathcal{C}, \mathcal{R}, \mathcal{P}$
Function $Z = \textbf{FFT\_Estimate}(\mathcal{P}, \mathcal{C}, \mathcal{R}, M)$
$d \leftarrow \texttt{ndims}(\mathcal{P})$ *Get number of dimensions*
$\mathcal{P}_f \leftarrow \text{FFT}_d(\mathcal{P})$ *Diagonalize model*
$\mathcal{C}_f \leftarrow \text{FFT}_d(\mathcal{C})$
$\mathcal{R}_f \leftarrow \text{FFT}_d(\mathcal{R})$
$Z \leftarrow \texttt{real}\left[\text{FFT}_d^{-1}\left(\left(\mathcal{P}_f \,.*\, \mathcal{C}_f^T ./ \left(\mathcal{C}_f \,.*\, \mathcal{P}_f \,.*\, \mathcal{C}_f^T + \mathcal{R}_f\right)\right) .* \text{FFT}_d(M)\right)\right]$

Goals: Compute a random sample, size $\underline{n}$, from a stationary, periodic prior kernel
Function $Z = \textbf{FFT\_Sample}(\mathcal{P}, \underline{n})$
$W \leftarrow \texttt{randn}(\underline{n})$ *Generate white random field*
$d \leftarrow \texttt{ndims}(\mathcal{P})$ *Get number of dimensions*
if $d \neq \texttt{length}(\underline{n})$ **then**
`Error`('Inconsistent kernel and sample dimensions')
end if
$Z \leftarrow \texttt{real}\left[\text{FFT}_d^{-1}\left(\texttt{sqrt}\left(\texttt{real}\left(\text{FFT}_d(\mathcal{P})\right) .* \text{FFT}_d(W)\right)\right)\right]$

Goals: Find the matrix inverse of a stationary, periodic kernel
Function $\mathcal{A}^{-1} = \textbf{FFT\_Inverse}(\mathcal{A})$
$d \leftarrow \texttt{ndims}(\mathcal{A})$ *Get number of dimensions*
$\mathcal{A}^{-1} \leftarrow \texttt{real}\left[\text{FFT}_d^{-1}\left(1./\texttt{real}\left(\text{FFT}_d(\mathcal{A})\right)\right)\right]$

then from (3.112), and ignoring prior means, we have

$$\hat{Z} = \text{FFT}_d^{-1}\left\{\left(\bar{\mathcal{P}} \odot \bar{\mathcal{C}}^T \oslash \left(\bar{\mathcal{C}} \odot \bar{\mathcal{P}} \odot \bar{\mathcal{C}}^T + \bar{\mathcal{R}}\right)\right) \odot \text{FFT}_d(M)\right\} \tag{8.80}$$

$$\tilde{\mathcal{P}} = \text{FFT}_d^{-1}\left\{\bar{\mathcal{P}} - \bar{\mathcal{P}} \odot \bar{\mathcal{C}}^T \oslash \left(\bar{\mathcal{C}} \odot \bar{\mathcal{P}} \odot \bar{\mathcal{C}}^T + \bar{\mathcal{R}}\right) \odot \bar{\mathcal{C}} \odot \bar{\mathcal{P}}\right\}. \tag{8.81}$$

The posterior sampler follows from $\tilde{\mathcal{P}}$ in the same manner as (8.78):

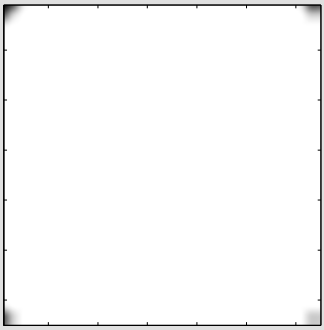
$$(Z|M) = \hat{Z} + \text{FFT}_d^{-1}\left(\sqrt{\text{FFT}_d(\tilde{\mathcal{P}})} \odot \text{FFT}_d(W)\right). \tag{8.82}$$

Example 8.3: FFT and Spatial Statistics

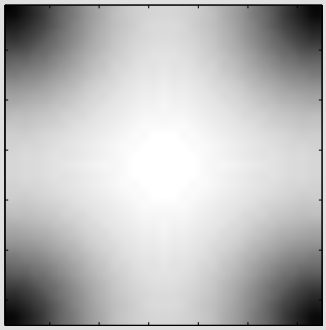
We use the FFT method to examine the thin-plate prior model of Figure 5.8:

$$\begin{array}{ccccc} & & 1 & & \\ & 2 & -8 & 2 & \\ 1 & -8 & \boxed{20.001} & -8 & 1 \\ & 2 & -8 & 2 & \\ & & 1 & & \end{array}$$

To use this kernel as the prior to an $N \times N$ image we need to embed the above values in an $N \times N$ periodic kernel $\mathcal{Q} = \mathcal{P}^{-1}$, with the kernel origin in the upper left corner, from which the correlation kernel is easily found via (8.72):

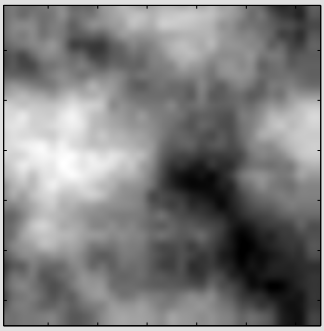


Thin-plate model kernel $\mathcal{P}^{-1}$

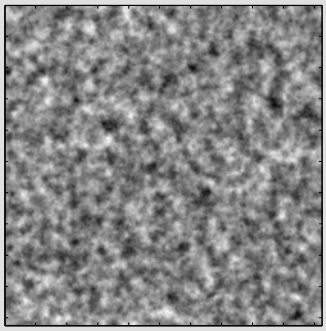


Thin-plate correlation kernel $\mathcal{P}$

With the correlation kernel in place, it is trivial to use (8.78) to generate random prior samples. Because of the efficiency of the FFT, generating very large random fields (right) is easy. The left sample shows particularly clearly the periodicity (top/bottom and left/right) of the field.



Low-resolution prior sample



High-resolution prior sample

8.4 Hierarchical Bases and Preconditioners

The previous sections have examined basis reduction, to reduce problem size, and the highly specialized approach of using the FFT for problem diagonalization.

We now return to the question of whether a general-purpose change of basis can be found, appropriate for spatial estimation. We recall from Section 8.1 that for sparsely measured domains an explicit change of basis may not be appropriate, rather an implicit reformulation of the estimation problem is found by transforming the linear system

$$\begin{aligned} \hat{\underline{z}} = \left(P^{-1} + C^T R^{-1} C\right)^{-1} C^T R^{-1} \underline{m} \quad &\Rightarrow \quad \left(P^{-1} + C^T R^{-1} C\right) \hat{\underline{z}} = C^T R^{-1} \underline{m} \\ &\Rightarrow \quad A \hat{\underline{z}} = \underline{b} \end{aligned} \tag{8.83}$$

by a change of basis $\underline{z} = S\bar{\underline{z}}$:

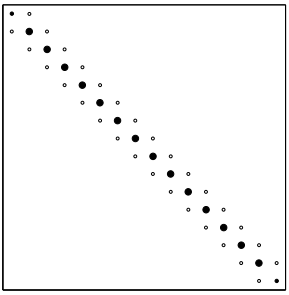
$$\begin{array}{ccccc} A\underline{z} = \underline{b} & \longrightarrow & S^T A\, S\bar{\underline{z}} = S^T \underline{b} & \longrightarrow & \bar{A}\bar{\underline{z}} = \bar{\underline{b}} \\ & & & & \downarrow \text{ Easy ?} \\ \hat{\underline{z}} & \longleftarrow & \text{S} & & \hat{\bar{\underline{z}}} \end{array} \tag{8.84}$$

Recall from Figures 8.1 (page 242) and 8.2 that the poor conditioning of A, especially in our context of random fields, stems from interaction locality.

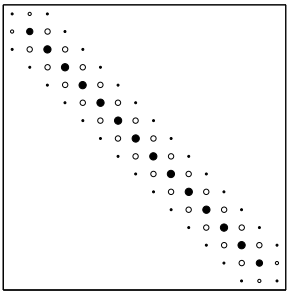
In principal components, a change of basis was inferred from the problem covariance, but made no assumptions regarding the arrangement of state elements in $\underline{z}$. Now, in contrast, we wish to explicitly recognize that $\underline{z}$ represents a spatial problem with the assumption that nearby pixels interact strongly. Can we introduce a change of basis S in which the basis elements $\bar{\underline{z}}$ are nonlocal, allowing spatially-separated state elements to be coupled?

Even in the spatial case, the perfect change of basis remains the eigendecomposition which will, in general, have highly nonlocal basis vectors. However, we do *not* want to precondition with a set of highly nonlocal vectors, because it is unlikely that we will happen to select something that resembles eigenvectors, and indeed highly possible that we may create a set of coefficients even *more* correlated than before.

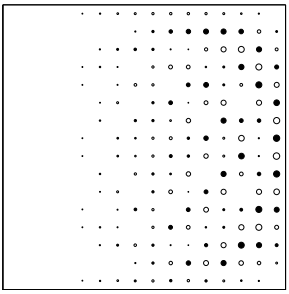
That a nonlocal basis $\bar{\underline{z}}$ is not, on its own, enough to necessarily improve the conditioning of $\bar{A}$ over A can be seen in Figure 8.12, which illustrates the effect of five preconditioners on the conditioning of a first- and a second-order system. The local averaging operator, in which each element in the transformed space is a smeared or locally-averaged version of the original space, is a nonlocal operator, however the conditioning was made *worse* for the first-order test case.



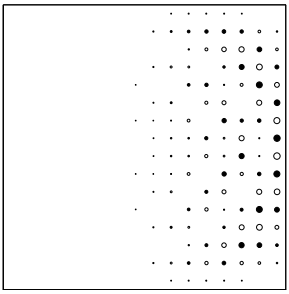
$A_1 = \left(C^T R^{-1} C + \lambda L^T L\right)$ for First-Order L



$A_2 = \left(C^T R^{-1} C + \lambda L^T L\right)$ for Second-Order L

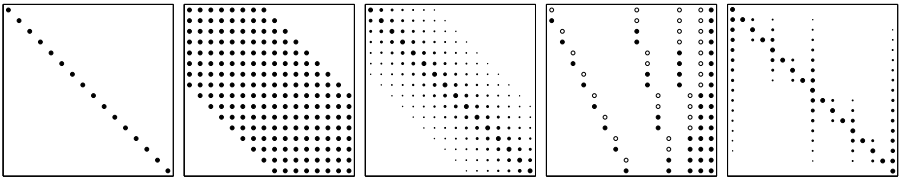


Ideal Preconditioner for A_1



Ideal Preconditioner for A_2

The columns (eigenvectors) in the ideal preconditioners are weighted by the eigenvalues to show significance. For a given change of basis matrix S, we can compute the problem conditioning as $\kappa_i = \kappa(S^T A_i S)$:

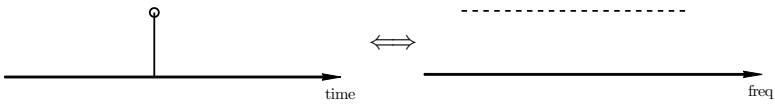


No Preconditioning	Local Average	Weighted Average	Hier. – Wavelet	Hier. – Triangular
$\kappa_1 = 39$	$\kappa_1 = 154$	$\kappa_1 = 26$	$\kappa_1 = 8$	$\kappa_1 = 5$
$\kappa_2 = 1485$	$\kappa_2 = 939$	$\kappa_2 = 3$	$\kappa_2 = 284$	$\kappa_2 = 66$

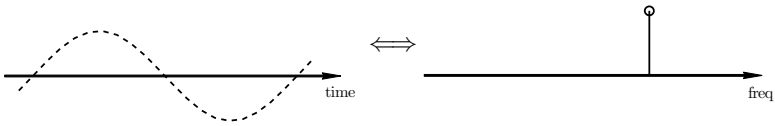
Fig. 8.12. Illustration of preconditioning: Each panel plots the sparsity structure of the corresponding matrix, with circle size related to magnitude, and the filled/unfilled state related to sign. It is clear that problem conditioning can change greatly as a function of preconditioning, although the best choice of preconditioner will vary with the problem.

Let us briefly consider the most obvious forms of nonlocal behaviour. As is very familiar from signal analysis [244], there are two extremes of local / nonlocal behaviour:

Impulse:

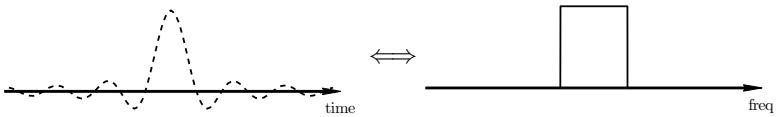


Sinusoid:

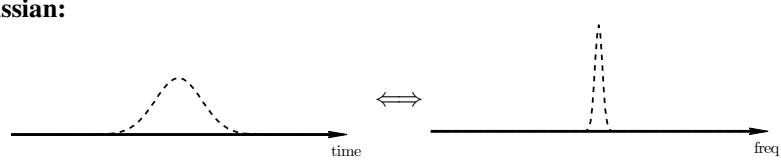


Because we are trying to find basis vectors which are not purely local, but at the same time not highly nonlocal, we are motivated to consider vectors at a more intermediate scale, localized in both the time and frequency domains:

Sinc:



Gaussian:



The latter Gaussian function is particularly interesting for being local in both the spatial and frequency domains.

It is possible to create bases using shifted versions of all of the above functions, however empirically it has been found that effective preconditioners need to introduce a *range* of nonlocality, such that both local and nonlocal effects can be effectively represented in the transformed space. Unfortunately, it is *not* obvious how to take shifted versions of the above functions at various scales to make a multiresolution basis.

Hierarchical systems offer precisely such a range of nonlocality, and two such hierarchical bases are discussed in the following sections.

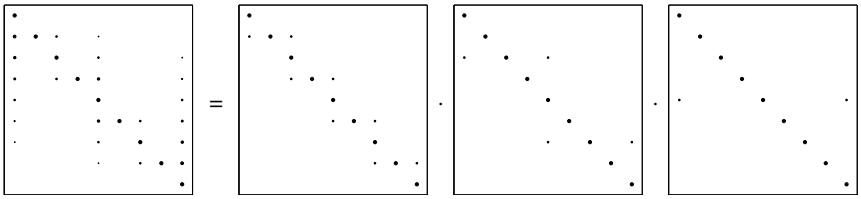


Fig. 8.13. A hierarchical interpolant S can be built up as the product of local interpolations S_j over scales. Here S_0 interpolates the midpoint from the two endpoints, with progressively local interpolants in S_1 and S_2.

8.4.1 Interpolated Hierarchical Bases

The simplest approach to creating a basis having a range of scales is to develop a hierarchical interpolator. Precisely such a method comes out of the preconditioning literature [347,348] and has been used for multidimensional surface estimation [298, 299].

If we are wishing to specify a hierarchical interpolator, it is cleaner, more intuitive, and possibly much simpler to construct S hierarchically, rather than as a single transformation. Therefore our goal is to write the transformation as

$$\underline{z} = S\bar{\underline{z}} = S_J S_{J-1} \cdots S_1 S_0 \bar{\underline{z}} \tag{8.85}$$

over J scales, as illustrated in Figure 8.13, such that S_j represents the interpolation at the jth scale. At each scale, the new nodes introduced at a scale are interpolated from nodes at the coarser scale, a simple operation which leads to very sparse and simply-structured S_j.

The resulting change of basis for a three-scale representation of a one-dimensional process gives a total of nine basis elements, implied by the columns of S in Figure 8.13, and plotted graphically in Figure 8.14.

We do not, however, wish to specify S explicitly: for a $N \times N$ domain, the matrix S will be $N^2 \times N^2$, a very large matrix to store, even if sparse. As in the lengthy discussion of sparse kernels in Section 5.3.2, a similar conclusion applies: we should specify S *implicitly*, via an interpolation algorithm

$$\underline{q} = S_j \bar{\underline{q}} \equiv \text{Interp}(\bar{\underline{q}}, j) \tag{8.86}$$

so that the net transformation is computed as

$$\underline{z} = \text{Interp}\Big(\text{Interp}\big(\ldots \text{Interp}(\bar{\underline{z}}, 0) \ldots, J-1\big), J\Big). \tag{8.87}$$

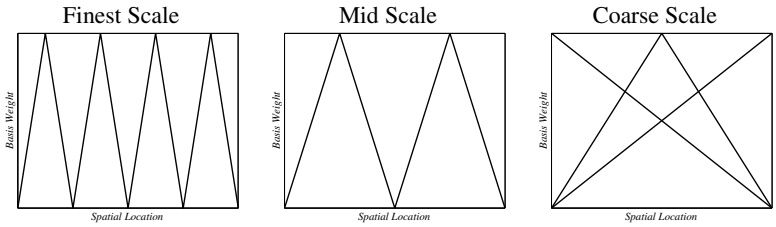


Fig. 8.14. The hierarchical interpolator S in Figure 8.13 is composed of nine interpolants (the columns of S) at different scales. If local interpolation is used in S_j, then the resulting interpolants in S are triangular, as shown.

The implicit, algorithmic formulation of (8.87) facilitates the generalization of this approach to methods other than one-dimensional triangular. In particular, the interpolation may be chosen to be linear or nonlinear (cubic etc.); similarly if $\underline{z}$ is understood to be a multidimensional field, organized lexicographically, then the interpolation can just as easily be bilinear or bicubic.

The only disadvantage of this interpolative method is the need to specify and implement it in the first place. Given the widespread availability of implemented wavelet transforms, we are highly motivated to consider wavelets, discussed in the following section.

8.4.2 Wavelet Hierarchical Bases

A multiresolution decomposition [221, 293] is one in which an interpolator

$$A_j : V_\infty \to V_j \tag{8.88}$$

maps from infinite spatial resolution V_∞ to a vector space V_j at resolution j. Under the conditions that

1. A coarse resolution signal is representable at finer resolutions:

$$V_j \subset V_{j+1} \subset \cdots \tag{8.89}$$

2. The subspaces are stretched by a factor of two:

$$f(t) \in V_j \iff f(2t) \in V_{j+1}. \tag{8.90}$$

3. Shift invariance

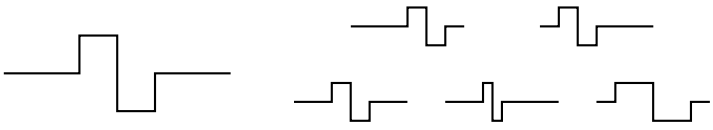


Fig. 8.15. The Haar wavelet basis: The single Haar wavelet function, left, forms a basis when shifted and dilated, right.

then[6] there exists a *unique* function ϕ such that

$$\left\{2^j \phi(2^j t - n)\right\} \tag{8.91}$$

is a basis for V_j. However this still leaves us with a basis in which all elements are at one scale, like a set of shifted *sinc* or *Gaussian* functions. Furthermore, we cannot just combine basis elements from different scales j, since that may not leave us with a basis.

Instead, the key idea with wavelets is the following:

> *Suppose we have a basis for a coarse resolution. What* **new** *information do we need in order to represent a finer scale?*

Let us seek to define a vector space W_j such that

$$V_{j+1} = V_j \oplus W_j \qquad V_j \perp W_j \tag{8.92}$$

meaning that

$$\left.\begin{array}{l}\text{Basis for coarse scale } V_j \\ \text{Basis for missing details } W_j\end{array}\right\} \text{ Basis for finer scale } V_{j+1}. \tag{8.93}$$

Under the conditions [221] for the multiresolution decomposition, there exists a *unique* function ψ such that shifted and dilated versions of ψ

$$\left\{2^j \psi(2^j t - n)\right\} \tag{8.94}$$

form a basis for W_j, as is illustrated in Figure 8.15 for the simple case of the Haar wavelet. By extending the decomposition of (8.92),

$$\begin{aligned} V_{j+1} &= V_j \oplus W_j && (8.95)\\ &= \left(V_{j-1} \oplus W_{j-1}\right) \oplus W_j && (8.96)\\ &= \underbrace{V_{j-q}}_{\text{Very Coarse}} \oplus \underbrace{W_{j-q}}_{\text{Coarse}} \oplus \cdots \oplus \underbrace{W_j}_{\text{Very Fine}}, && (8.97)\end{aligned}$$

[6] There are technical conditions that $\cup_j V_j$ is dense and $\cap_j V_j = \{0\}$.

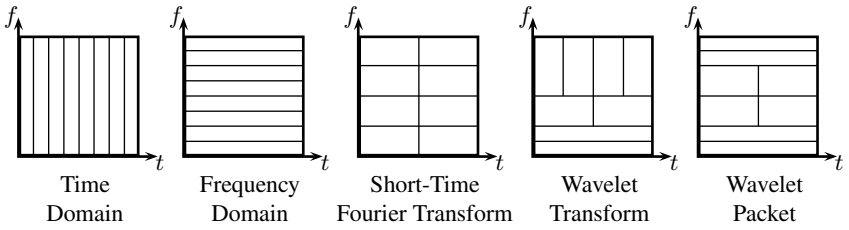


Fig. 8.16. The space (or time) and frequency domains can be carved up in various ways. The figure shows five possible representations of eight coefficients. The wavelet approaches are unique, offering high-resolution coefficients in both the temporal and the frequency domains.

we observe that we have met our goal of expressing a basis for V_{j+1} in terms of basis elements at all scales. The resulting wavelet basis essentially represents a tradeoff between the time and frequency representations, as illustrated in Figure 8.16.

The generalization of wavelets to multiple dimensions is straightforward, both for computational and implementation reasons. The $\mathcal{O}(n)$ complexity of the wavelet transform is fast, as opposed to the $\mathcal{O}(n \log n)$ complexity of the fast Fourier transform, and is therefore applicable to exceptionally large problems. Furthermore, for orthogonal wavelets, the multidimensional transform can be computed by taking one-dimensional transforms along each dimension, as with the Fourier transform.

The basis change for a multidimensional, lexicographically ordered $\underline{z}$ is thus easily accomplished by using the corresponding multidimensional wavelet transform [345]

$$\underline{z} = \mathrm{WT}(\tilde{\underline{z}}). \tag{8.98}$$

The orthogonality of the wavelet transform, like the Fourier transform, means that the inverse and adjoint (transpose) operators are equivalent, such that the product $\bar{A}\underline{\tilde{z}}$ from (8.7) is easily computed as

$$\bar{A}\underline{\tilde{z}} = S^T A S \underline{\tilde{z}} \tag{8.99}$$

$$= \mathrm{WT}^{-1}\Big(A \cdot \mathrm{WT}\big(\underline{\tilde{z}}\big)\Big). \tag{8.100}$$

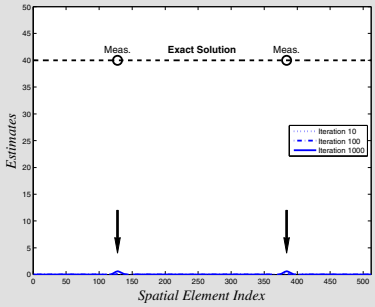
8.4.3 Wavelets and Statistics

The great many choices of wavelets and the computational efficiency and hierarchical representation of the wavelet transform have led to extensive study, including some work on wavelet statistical properties [237, 317].

Example 8.4: Hierarchical Bases

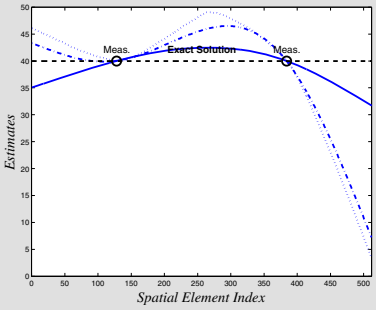
Suppose we consider a one-dimensional, second-order interpolation. We will apply a simple, iterative approach (Gauss–Seidel, of Section 9.2) solution to the resulting linear system. The second-order (thin plate) problem is badly conditioned, such that after one thousand iterations, almost no progress has been made towards the desired solution:

No Preconditioning

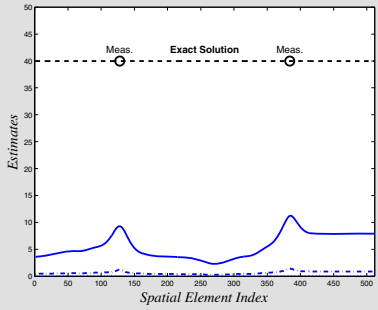


In contrast, the hierarchical preconditioned problems converge much faster. Because thin plate implies a rather smooth prior, the smoother bases (triangular, DB4) converge more rapidly, with DB4 mostly converged in only ten iterations:

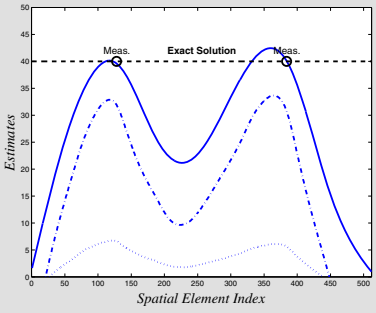
Hierarchical Triangular



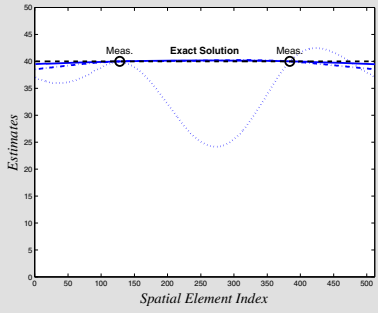
Haar Wavelet (DB1)



Daubechies Wavelet DB2



Daubechies Wavelet DB4



Given a random field or process $\underline{z} \sim P$, the transformation of this process by a wavelet transform leads to the modified statistics

$$\bar{\underline{z}} = \mathrm{WT}(\underline{z}) = W\underline{z} \quad \Rightarrow \quad \bar{\underline{z}} \sim \bar{P} = WPW^T. \tag{8.101}$$

In many cases, certainly for many smooth $\underline{z}$ but also for fractal fields [117], $\bar{P}$ is assumed to be nearly diagonal, meaning that $\underline{z}$ has been decorrelated. Indeed, that the wavelet transform nearly whitens many signals is precisely the rationale for its use in preconditioning and changes of basis.

Because images are usually densely measured, the near-whitening property of the wavelet transform has made it popular as an *explicit* change of basis in image processing. If a process is densely measured with white additive noise,

$$\underline{m} = I\underline{z} + \underline{v} \qquad \underline{v} \sim \sigma I \tag{8.102}$$

then the whole problem can be transformed, using an orthogonal wavelet W, into the wavelet domain:

$$\bar{\underline{m}} = W\underline{m},\ \bar{\underline{z}} = W\underline{z},\ \bar{\underline{v}} = W\underline{v} \quad \Rightarrow \quad \bar{\underline{m}} = I\bar{\underline{z}} + \bar{\underline{v}} \qquad \bar{\underline{v}} \sim \sigma I \tag{8.103}$$

such that the transformed problem has two particularly convenient properties:

1. The noise $\bar{\underline{v}}$ remains white, because of the orthogonality of W.
2. The prior model for $\bar{\underline{z}}$ may be chosen to be diagonal.

Therefore the resulting estimation problem is pointwise, without spatial interactions, which has led to a wide variety of wavelet methods for image estimation [114, 115].

The whitening behaviour of most wavelets implies that the wavelet transform of real images is sparse, meaning that most of the coefficients are near zero. This observation has led to a variety of image-compression methods [302]. Similarly, the sparsity of the coefficients has allowed the statistical behaviour of the nonzero coefficients to be studied, particularly for basic fundamental image shapes (steps, edges, etc.), which has led to methods for wavelet-based image resolution enhancement [262].

The marginal statistics for $\underline{z}'$ are not, however, Gaussian. For two-dimensional pictures, the marginal statistics of $\underline{z}'$ are more strongly concentrated at zero than a Gaussian, and the wavelet coefficients have been modelled as Rician, generalized Gaussian [57], and Gaussian scale mixtures [263] among others. Because the marginal statistics are no longer Gaussian, the optimal Bayesian estimator is nonlinear, which has led to a wide variety of nonlinear and thresholding approaches to image denoising [56, 87], a simple version of which is explored in Problem 8.5.

The wavelet transform is not, of course, a perfect whitener. The spatial correlations have been studied [13, 306], although more success has been had by studying the

parent–child relationship of coefficients [77,149,262,274], which led to models such as the Hidden Markov Tree, which was illustrated in Figure 7.7.

The hierarchical nature of wavelets also makes them a natural choice for the analysis and representation of power-law or $1/f$ processes [339, 340], and which has been used as the basis for estimating power-law fields [105].

8.5 Basis Changes and Markov Random Fields

Given the interest throughout this text in Markov random fields and sparse models, it is worthwhile asking how such models are affected by a change of basis, and particularly by a hierarchical change.

In general, given a sparse random field Z, such that

$$[Z]_{:} = \underline{z} \sim P \qquad P^{-1} \text{ sparse} \tag{8.104}$$

then if this field is operated upon by some change-of-basis operator F, the transformed field

$$\bar{\underline{z}} = F\underline{z} \sim \bar{P} \qquad \bar{P}^{-1} = \left(FPF^T\right)^{-1} = F^{-T}P^{-1}F^{-1}, \tag{8.105}$$

where we assume that F is square and invertible. Since it is F which would normally be specified in a change of basis, and not F^{-1}, it is exceptionally unlikely that F^{-1} would in any way be sparse, therefore almost certainly the sparsity of P^{-1} has been lost in $\bar{P}^{-1}$.

This loss of sparsity is not necessarily tragic, however, since the transformed system may not need to be written down explicitly or, in the case of a reduction of basis, the transformed system may be sufficiently small that a sparse representation is not needed. Recalling the implicit change of basis from (8.84), we had the transformation of a linear system

$$A\underline{z} = \underline{b}, \tag{8.106}$$

where A is sparse, to the transformed system

$$S^TA\,S\bar{\underline{z}} = S^T\underline{b} \quad \Longleftrightarrow \quad \bar{A}\bar{\underline{z}} = \bar{\underline{b}}, \tag{8.107}$$

where $\bar{A}$ is now dense, as argued in the previous paragraph. However, we do not necessarily need to store $\bar{A}$; instead, the matrix–vector product $\bar{A}\bar{\underline{z}}$ is calculated as

$$\bar{A}\bar{\underline{z}} \equiv S^T\Big(A\big(S\bar{\underline{z}}\big)\Big). \tag{8.108}$$

That is, we require only the ability to compute the three matrix–vector products $S^T\underline{x}$, $A\underline{y}$, $S\underline{z}$ to preserve the benefits of the sparsity of A.

The above discussion dealt with changes and reductions of basis in general. However, there is a much more interesting, specific question:

Given a Markov random field Z, if I subsample the random field or represent it at a reduced resolution, is the resulting field still Markov?

Let us start with a few special cases. If our random field Z is zeroth-order Markov (white noise), then any orthogonal resolution reduction will result in a zeroth-order Markov field, since

$$\underline{z} \sim I \quad \Longrightarrow \quad \underline{\bar{z}} = F\underline{z} \sim FIF^T = FF^T = I. \tag{8.109}$$

However the absence of any spatial interactions in the zeroth-order case makes it of limited interest. Certainly if Z is first-order Markov, such that the columns of Z satisfy

$$E_{\text{LLSE}}\left[\underline{z}_{i-1}|\underline{z}_i,\ \underline{z}_{i+1},\ \underline{z}_{i+2},\ \ldots\right] = E_{\text{LLSE}}\left[\underline{z}_{i-1}|\underline{z}_i\right] \tag{8.110}$$

then it follows that

$$E_{\text{LLSE}}\left[\underline{z}_{i-2}|\underline{z}_i,\ \underline{z}_{i+2},\ \underline{z}_{i+4},\ \ldots\right] = E_{\text{LLSE}}\left[\underline{z}_{i-2}|\underline{z}_i\right], \tag{8.111}$$

meaning that if we subsample the columns of Z, the resulting random field is still column-wise Markov, although row-wise the Markovianity will (most likely) have been lost.

The loss of Markovianity in downsampling stems from the fact that the exact, fine-scale state values are lost, and replaced with functions (e.g., averages) of those state values. Consider, for example, a Markov process with four state elements

$$\underline{z}^T = \boxed{z_1}\boxed{z_2}\boxed{z_3}\boxed{z_4} \tag{8.112}$$

Suppose the process is first-order Markov, meaning that either z_2 or z_3 can conditionally decouple z_1 and z_4. The process prior inverse could look something like

$$P^{-1} = G = \begin{bmatrix} 1.2 & -1 & 0 & 0 \\ -1 & 2.2 & -1 & 0 \\ 0 & -1 & 2.2 & -1 \\ 0 & 0 & -1 & 1.2 \end{bmatrix} \tag{8.113}$$

Those entries with the grey background highlight those parts of G which must be zero in order for the prior to be first-order Markov.

Now suppose we modify the process by averaging the middle two elements:

$$(F\underline{z})^T = \underline{\bar{z}}^T = \boxed{z_1}\boxed{\tfrac{z_2+z_3}{2}}\boxed{z_4} \tag{8.114}$$

where F is a 3×4 array. The new middle element contains the information from both neighbouring values in the middle of the original process; we might näively suppose that

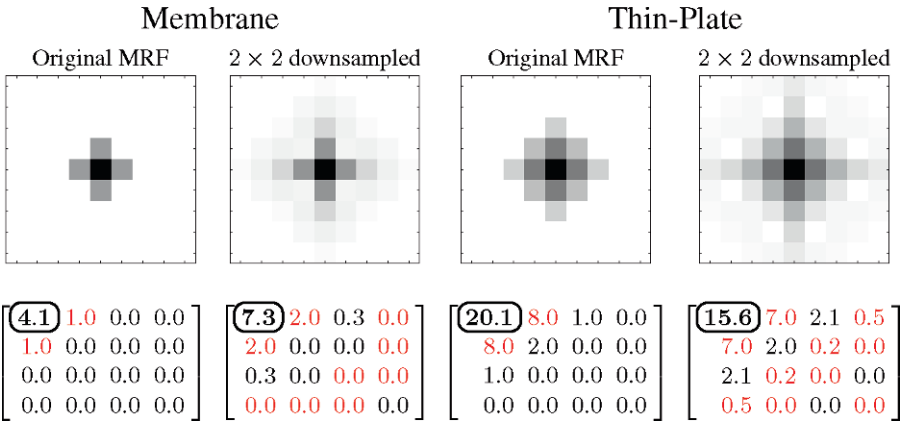


Fig. 8.17. The panels show two examples of downsampling two-dimensional MRFs. Random fields corresponding to both kernels are created in a 2D domain, which is subsampled by 2×2 block averaging. Although the subsampled field is no longer Markov, it is clear that the subsampled field is *very nearly* Markov, based on the sparsity of the subsampled kernel. It should come as no surprise that the kernel values have changed, since subsampling affects the random field variance and correlation structure. Both priors illustrated here are isotropic, therefore only one quadrant of the kernel is shown, with the kernel origin in the upper-left corner.

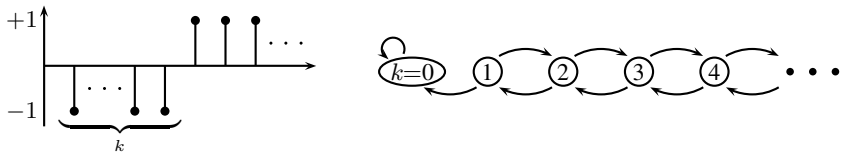
$$\frac{z_2 + z_3}{2}$$

represents a boundary of thickness two, and therefore easily conditionally separating z_1 and z_4. However, if we examine the prior inverse of $\underline{\bar{z}}$,

$$\bar{P}^{-1} = \left(FG^{-1}F^T\right)^{-1} = \begin{bmatrix} 1 & -1 & 0.16 \\ -1 & 2.4 & -1 \\ 0.16 & -1 & 1 \end{bmatrix} \tag{8.115}$$

we find that this process is, indeed, no longer first-order Markov, by the presence of nonzero values in the greyed entries. We do notice, however, that the greyed entries are small. Indeed, Figure 8.17 shows the kernels which arise from downsampling two-dimensional MRFs; the downsampled fields, although not precisely Markov, are very nearly so. In many cases it may be very reasonable to approximate a downsampled MRF as Markov.

It is possible to consider the question of constructing exact Gibbs/Markov process at multiple resolutions [136], however the results are very complicated, in part because the downsampled field is not Markov and therefore gives rise to nonlocal cliques and neighbourhoods. The question of downsampling and subsampling MRFs has



(a) 1D Ising Process (b) Markov Chain for the Random Walk in k

Fig. 8.18. In a one-dimensional Ising process (left) the boundary, dashed, between two state values will move as a random walk (right), until the optimum is reached when $k = 0$. The details of the transition probabilities in the corresponding Markov chain will depend on the details of the sampling algorithm.

been examined in detail in [199], and also in [146, 197, 199, 254]. There has also been considerable work in performing image operations, such as segmentation or denoising, using hierarchies of Markov random fields [35,58,146,186,234,334,341].

8.6 Basis Changes and Discrete-State Fields

Most discrete-state models, such as those discussed in Section 7.4, are *local* Markov / Gibbs, and therefore are subject to the same issues of indirection as local continuous-state models, as was discussed at the start of this chapter and illustrated in Figure 8.1 on page 242.

Although the same notion of ill-conditioning does not apply to discrete-state random fields, the slowness of convergence seen in Figure 8.2 applies equally to the discrete-state case, as was discussed in the image segmentation illustration of Application 7 on page 232.

Figure 8.18(a) illustrates the locality principle for the discrete-state Ising model (Section 7.4.1). Given a one-dimensional binary process, with k elements of value -1 followed by an indefinite sequence of $+1$, the probability of the Ising prior is clearly maximized when all of the states have the same value, meaning that the k elements point up. Under certain assumptions[7] the only state elements that can change are those on either side of the dashed boundary, leading the value of k to undergo a random walk over time, as illustrated in Figure 8.18(b), a well-known Markov chain (the "Gambler's Ruin" problem [248]). The optimum, at $k = 0$, is eventually reached, but the number of iterations is quadratic in k.

[7] The sampling of such processes will be discussed in Section 11.3; we are assuming a Gibbs sampler with a random site-visiting scheme at temperature $T = 0$.

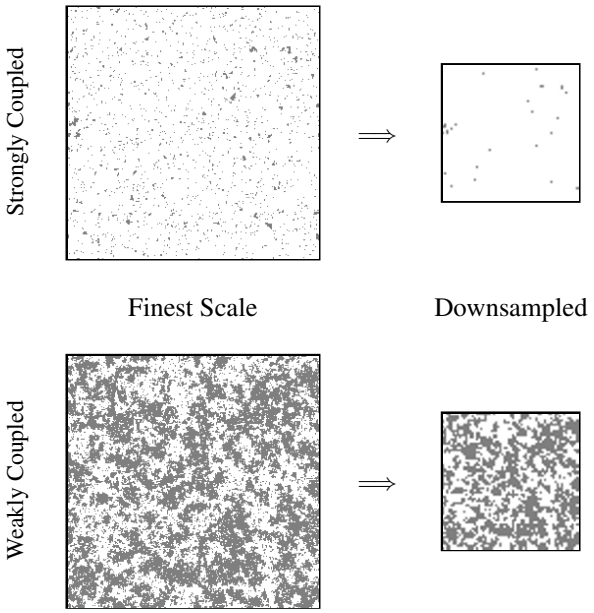


Fig. 8.19. The downsampling of Ising models: strongly coupled fields become more strongly coupled, top, and weakly coupled fields more weak, bottom.

Consequently, for local random field models it is exceptionally difficult to synthesize structures large in size relative to the local neighbourhood.

In response to this observation, we are motivated to consider changes of basis for discrete-valued random fields. The key difficulty is that discrete state fields do not allow a tapered basis. That is, essentially all of the continuous state bases — spatial functions in Figure 8.8, overlapped methods in Figure 8.9, hierarchical triangles in Section 8.4.1, wavelets in Section 8.4.2 — rely on a large-scale basis element being spatially tapered. In the continuous-state case, a large-scale element can be a smooth, low-resolution representation, which is then incrementally nudged and refined towards finer scales. However, a discrete hierarchy does not allow for a smoothly-varying representation or for small refinements in state value from scale to scale.

Fundamentally, the issue is that discrete fields are not closed under addition or multiplication, meaning that one cannot even write $\bar{\underline{z}} = F\underline{z}$, as in (8.1), for discrete $\underline{z}$. That is, a literal change of basis will not be possible.

It is, however, possible to construct discrete-state hierarchies based on downsampling. For example, given an energy function $H(\underline{z})$, we can seek to find the energy $H^k(\downarrow^k \underline{z})$ corresponding to the field after k scales of downsampling. It is possible to

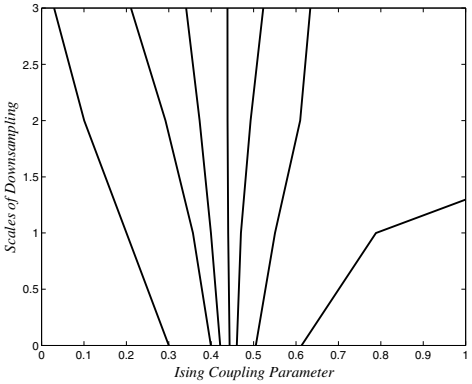


Fig. 8.20. Downsampling a 2D Ising model exaggerates the difference between the coupling β and the critical value of 0.4407, as was illustrated for two cases in Figure 8.19.

derive H^k analytically, per the seminal work by Gidas [136], however in practice it is common to assume the *form* of the energy function,

$$H^k(\downarrow^k \underline{z}) \equiv H(\downarrow^k \underline{z}, \theta_k) \tag{8.116}$$

with the scale dependence only in model parameter θ. The famous example of such downsampling is the Ising model (Section 7.4.1) [26, 171, 335] which, after downsampling, remains an Ising model but with a different degree of coupling:

- A *weakly* coupled field possesses only small structures. When downsampled, the structures become even smaller, thus closer to random
  $\Longrightarrow$ Even more weakly coupled
- A *strongly* coupled field has most pixels with the same state value, with occasional outliers. When downsampled, outliers tend to be removed
  $\Longrightarrow$ Even more strongly coupled

as illustrated for two examples in Figure 8.19, and plotted as a function of β in Figure 8.20.

The intent of the Ising examples is to illustrate that a hierarchical representation, even if not a change of basis, is plausible for discrete-state fields. There are two hierarchies to consider:

1. **Top-down,** in which the hierarchy begins at a coarse scale, with the coarse-scale elements repeatedly refined at finer scales.

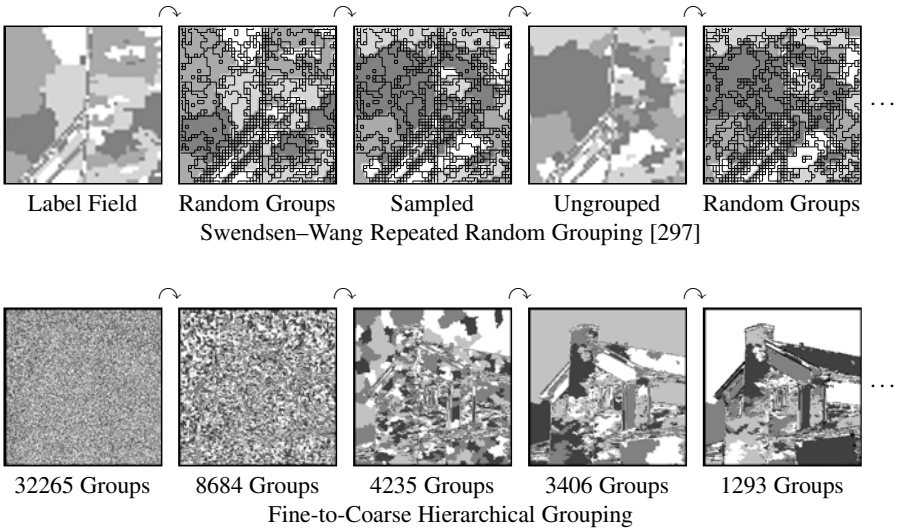


Fig. 8.21. Two approaches to discrete-state grouping. Groups may be repeatedly created and destroyed, top, or groups may persist, bottom, and be grouped further. The hierarchical grouping leads to much larger groups and better computational efficiency, but may suffer from the persistence of a poor grouping.

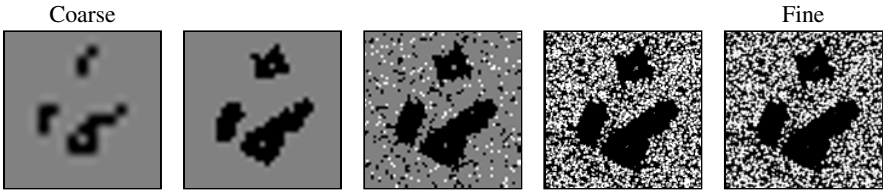
2. **Bottom-up,** in which the hierarchy begins at the finest, pixellated scale, and where some sort of grouping or aggregation leads to coarser representations.

For discrete-state problems, the bottom-up approach is considerably more common, precisely because of the ambiguity in representing a discrete field at a coarse scale. Because the field is discrete, however, it will be common to have adjacent elements with precisely the same value, leading to a natural notion of grouping, as illustrated in Figure 8.21. The Swendsen–Wang method [184, 297] was proposed for the acceleration of Ising–Potts models, whereby groups of state elements are changed simultaneously, rather than one pixel at a time. Because the domain is periodically ungrouped and regrouped, no grouping assignment is ever fixed, meaning that a poor grouping at some iteration does not compromise the final result.

In contrast, methods of region grouping [217, 314, 329] lead to much more rapid convergence by growing much larger groups, but with the cost that grouping cannot be undone, meaning that grouping errors persist in future iterations.

Top-down approaches, which dominate in continuous-state problems, can also be applied in the discrete-state case, as shown in Figure 8.22. In a regular coarse-to-fine hierarchy [5], the result from a coarser scale becomes the initialization at a finer

Coarse-to-Fine Binary Hierarchy [5]



Coarse-to-Fine Ternary Hierarchy, Sampling Only Grey Elements [50]

Fig. 8.22. Two approaches to discrete-state top-down hierarchies. The result at each scale may only serve as the initialization of the next, top, or each scale may constrain the behaviour of the next, bottom.

scale, leading to the obvious problem that the fine scales are not prevented from diverging from the coarse-scale initialization.

Although not literally a change of basis, the ternary approach [50] illustrated in the bottom panel of Figure 8.22 is close in spirit to an orthogonal representation. The binary state is augmented to ternary by adding an undecided condition; all *decided* elements (black or white) are fixed, such that at any scale the large-scale structure inherited from coarser scales is unchangeable, and only the undecided elements are sampled. The small fraction of undecided pixels at fine scales leads to computational benefits similar to those of basis reduction.

Application 8: Global Data Assimilation [111]

Let us consider an example, building on the state-reduction methods described in Section 8.2.2.

Suppose we wish to simulate a General Circulation Model, a model of the world's oceans which is driven by observed data, such as sea-surface temperature. A good representation of the partial differential equations governing water circulation require a very *fine* resolution, however running a Kalman-like filter and producing

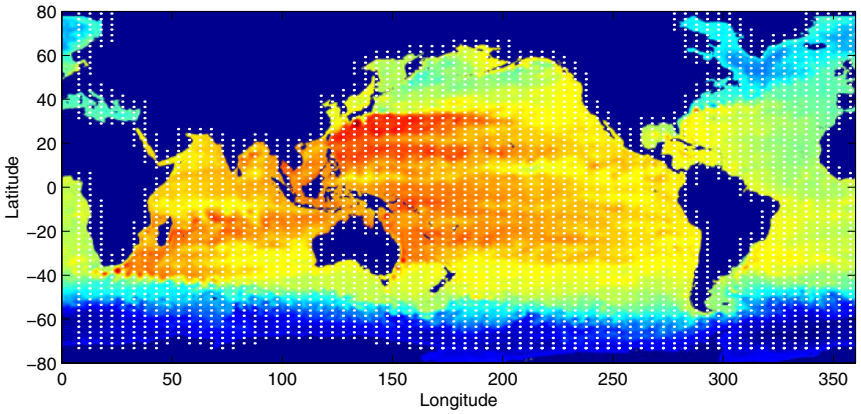


Fig. 8.23. Mapping test for global-scale problem, from [111]. The coarse grid is 71×62, superimposed on a 2160×960 fine grid. The centred locations of the 3551 interpolants are shown as white dots.

error statistics can be done only at a relatively *coarse* scale, hence we have a state-reduction or two-scale problem.

Specifically, suppose we have the global problem illustrated in Figure 8.23. The prediction step is performed by some numerical differential equation

$$\underline{z}(t+1|t) = \mathcal{A}\big(\underline{z}(t|t)\big). \tag{8.117}$$

The error statistics are updated using a reduced-order Kalman filter (Section 10.2.6), which operates on a much smaller state $\underline{\bar{z}}$,

$$\underline{\bar{z}}(t|t) = \mathcal{K}\big(\underline{z}(t|t-1)\big), \tag{8.118}$$

where the number and locations of the reduced states are indicated by the white dots in Figure 8.23.

The key challenge is therefore the transformation between the fine $\underline{z}$ and coarse $\underline{\bar{z}}$ scales, as in (8.8):

$$\underline{\bar{z}} = F\underline{z} \qquad \underline{z} = S\underline{\bar{z}}, \tag{8.119}$$

in particular, such that the transformation does not distort the coarse–fine–coarse cycle, meaning that we wish the pseudoinverse criterion

$$F\big(S(\underline{\bar{z}})\big) \equiv \underline{\bar{z}} \tag{8.120}$$

to be satisfied, exactly as discussed in Section 8.2. Figure 8.24 shows one fine–coarse–fine mapping, using a Gaussian kernel from Figure 8.8.

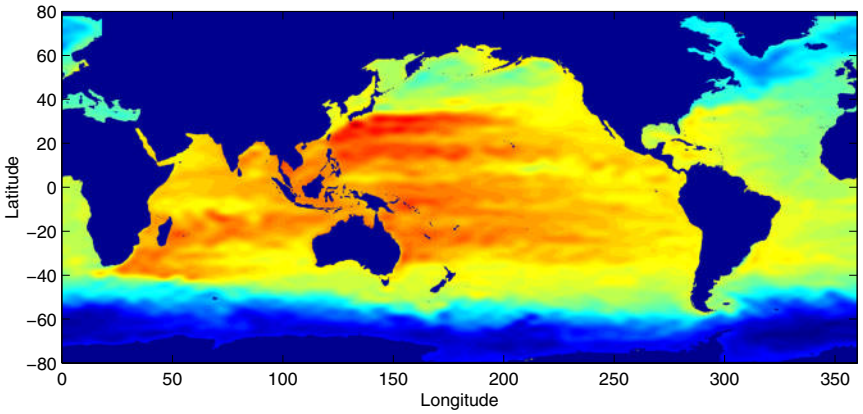


Fig. 8.24. A fine–coarse–fine mapping from Figure 8.23 [111], meaning that this fine-scale image was reconstructed from only 3551 coarse-scale values, a compression from the fine scale by a factor of over 400. Observe the absence of distortions, even along boundaries.

Summary

The following table gives a quick overview of the methods developed in this chapter and the contexts to which they apply.

Reductions of Basis:

Section	Method	Basic Assumptions
8.2.1	Principal Components	Problem stationarity along some dimension
8.2.2	Fast Pseudoinverses	Spatial smoothness
8.2.3	Local Processing	Limited correlation, dense measurements

Changes of Basis:

Section	Method	Basic Assumptions
8.3	FFT	Full stationarity and periodicity
8.4.1	Hierarchical Triangles	None
8.4.2	Hierarchical Wavelets	None

For Further Study

For aspects of principal components, the text by Jolliffe [179] is recommended. For interested readers, the generalization of principal components to independent components analysis is discussed in the elegant text by Hyvärinen, Karhunen, and Oja [166].

The papers by Szeliski [299] and Yaou and Chang [345] look at the solving of two-dimensional inverse problems using triangular and wavelet bases, respectively.

This chapter has given only the most basic introduction to simple, orthogonal wavelets, whereas there are a great number of orthogonal, bi-orthogonal, non-orthogonal, complex, complete, and overcomplete wavelet variations from which to choose. The text by Strang and Nguyen [293] is an excellent place to start.

Sample Problems

Problem 8.1: FFT Sampling

(a) Use the FFT method to generate a random sample of each of the three kernels in Example 6.1.

Consider the thin-plate kernel in Example 6.1. Set the central element to one of $20.1, 20.01, 20.001, 20.0001$; for each of these four values do the following:

(b) Use the FFT method to generate a random sample.
(c) Use the FFT method to find the eigenvalues of the kernel, and from those infer the condition number.

Problem 8.2: FFT Estimation

Consider the thin-plate kernel in Example 6.1. Set the central element to one of $20.1, 20.01, 20.001, 20.0001$; for each of these four values do the following:

(a) Use the FFT method to generate a random sample.
(b) Add unit variance Gaussian noise to each state element.
(c) Use the FFT method to compute state estimates and estimation error variances.
(d) Comment on how and why the estimation error variance varies with the selected prior kernel.

Problem 8.3: Basis Reduction

Suppose we construct an estimation problem similar to the one in Figure 8.2: $\underline{z}$ is one-dimensional, with 200 elements, having a first-order (membrane) prior. Because the first-order prior is singular, we will add an additional constraint

$$\underline{l}^T = \begin{bmatrix} 1 & 1 & \cdots & 1 & 1 \end{bmatrix} \tag{8.121}$$

which penalizes the deviations of $\underline{z}$ from zero.

(a) Calculate the prior covariance P.
(b) Find the eigenvector associated with the largest eigenvalue of P.
(c) Find the eigenvector associated with the eigenvalue of P closest to zero.
(d) Describe briefly the behaviour of a reduced-order system, on the basis of the eigenvectors preserved (large eigenvalue) and rejected (small eigenvalue).

Problem 8.4: Basis Reduction

Construct a one-dimensional estimation problem, as in Problem 8.3, but for both first-order and second-order priors, with two measurements:

$$m_1 = z_{50} + v_1 \sim \mathcal{N}(1,1) \qquad m_2 = z_{150} + v_2 \sim \mathcal{N}(1,1). \tag{8.122}$$

Calculate the covariances P_1, P_2 for the first- and second-order problems, respectively, and find the sorted singular values $\sigma_{j,i}$ of P_j:

(a) Plot the singular values $\sigma_{1,i}$ and $\sigma_{2,i}$.
(b) Interpret the singular value plot in terms of the degree to which the basis can be reduced.
(c) Interpret the singular value plot in terms of the condition number of the original problem.

Problem 8.5: Open-Ended Real-Data Problem — Wavelet-Based Estimation

Let us apply the wavelet change of basis from (8.103) to image denoising, a process known as wavelet *shrinkage* [56, 87]. Find an image Z (from the Internet):

(a) Add Gaussian noise V to form measurements $M = Z + V$.
(b) Take the wavelet transform $\bar{M} = \mathrm{WT}(M)$, for example using `wavedec` in MATLAB with wavelet 'db4'.
(c) Our "estimator" will be a simple, nonlinear threshold:

$$\hat{\underline{\bar{z}}}_i = \begin{cases} 0 & |\underline{\bar{m}}_i| < \zeta \\ \underline{\bar{m}}_i & |\underline{\bar{m}}_i| \geq \zeta \end{cases} \tag{8.123}$$

(d) Take the inverse wavelet transform $\hat{Z} = \mathrm{WT}^{-1}(\hat{\tilde{Z}})$.
(e) Experiment with different choices for the settings under your control:
  (i) The type of noise: Gaussian, salt-and-pepper
  (ii) The type of wavelet: DB1, DB2, DB4, ...
  (iii) The type of estimator: hard thresholding, soft thresholding etc.
  (iv) The choice of threshold ζ

Part III

Methods and Algorithms

9

Linear Systems Estimation

One of the most fundamental equations in this book is the solution to the Bayesian linear least-squares estimator of (3.114):

$$\hat{\underline{z}}(\underline{m}) = \underline{\mu} + \left(P^{-1} + C^T R^{-1} C\right)^{-1} C^T R^{-1} (\underline{m} - C\underline{\mu}) \tag{9.1}$$

$$\tilde{P} = \mathrm{cov}(\tilde{\underline{z}}) = \left(P^{-1} + C^T R^{-1} C\right)^{-1}. \tag{9.2}$$

The presence of matrix inversions clearly limits the size of the problem (the number of elements in $\underline{z}$) to which these equations can be applied directly. The enormous significance of these equations, however, is that in *every* least-squares problem, whether static or dynamic, possibly in some reduced dimensional space or transformed basis (as in Chapter 8), (9.2) needs to be solved.

Although (9.2) makes explicit reference to Bayesian prior P, we could equally well be solving a non-Bayesian least-squares estimation problem

$$\hat{\underline{z}}(\underline{m}) = \left(L^T L + C^T R^{-1} C\right)^{-1} C^T R^{-1} \underline{m}. \tag{9.3}$$

As the two forms are algebraically equivalent (Section 3.2.4), and as the focus of this chapter is the algorithmic solution of an algebraic problem the Bayesian / non-Bayesian differences are not relevant here, so without loss of generality we consider the estimator

$$\hat{\underline{z}} = \left(Q + C^T R^{-1} C\right)^{-1} C^T R^{-1} \underline{m} \tag{9.4}$$

from which follows the linear system

$$\underbrace{\left(Q + C^T R^{-1} C\right)}_{A} \hat{\underline{z}} = \underbrace{\left(C^T R^{-1}\right) \underline{m}}_{\underline{b}}, \tag{9.5}$$

a system known as the *normal* equations. We are thus left to solve a regular linear system or its preconditioned equivalent (8.7)

$$A\hat{\underline{z}} = \underline{b} \qquad \text{or} \qquad S^T A\, S\hat{\tilde{\underline{z}}} = S^T \underline{b}. \tag{9.6}$$

P. Fieguth, *Statistical Image Processing and Multidimensional Modeling*, Information Science and Statistics, DOI 10.1007/978-1-4419-7294-1_9, © Springer Science+Business Media, LLC 2011

As A has the same size as P, it is infeasible to consider a dense storage of A for large problems, so a sparse (Section 5.3) or implicit representation of A is needed; in particular, we need efficient ways of computing the matrix–vector product

$$A\underline{\hat{z}} = Q\underline{\hat{z}} + C^T R^{-1} C\underline{\hat{z}}. \tag{9.7}$$

For example, given a stationary problem with uncorrelated point measurements, then $C = I, R = \text{Diag}(\underline{r})$, and Q has a kernel $\mathcal{Q}$, thus

$$A\underline{\hat{z}} = Q\underline{\hat{z}} + I^T \left(\text{Diag}(\underline{r})\right)^{-1} I\underline{\hat{z}} = \left[\mathcal{Q} * \hat{Z}\right]_{:} + \underline{\hat{z}} \oslash \underline{r}, \tag{9.8}$$

much faster and requiring vastly less storage than the explicit calculation with full matrix A. Even with a more complex problem, say a nonstationary Markov prior with correlated point measurements, then A is still sparse, with b bands, where b is a function of the Markov order of the prior and the correlation structure in R. In this case the product $A\underline{\hat{z}}$ is still efficient, although the storage requirements for A are b times the storage size of $\underline{\hat{z}}$. In many such nonstationary cases, for example the interpolation of Example 5.1, matrix A (or its components Q, C, R) may not be stored at all, but rather be represented implicitly by being generated dynamically by a computer algorithm:

$$A\underline{\hat{z}} \equiv \underline{a}(\underline{\hat{z}}). \tag{9.9}$$

This is particularly useful for problems in which most state elements are stationary, represented by a kernel, with a limited number of exceptions (e.g., boundary conditions) to be computed separately. Under the assumption that the exceptions are not computationally difficult to detect, the computational complexity of computing $\underline{a}(\underline{\hat{z}})$ should be the same as that for the sparse matrix–vector product $A\underline{\hat{z}}$.

For Bayesian estimation problems it is also possible [298] to solve $\tilde{P}$ using linear systems techniques:

$$\tilde{P} = \left(P^{-1} + C^T R^{-1} C\right)^{-1} \quad \Longrightarrow \quad \left(P^{-1} + C^T R^{-1} C\right) \tilde{P} = I. \tag{9.10}$$

Thus the columns of $\tilde{P} = \left[\underline{\tilde{p}}_0 \; \underline{\tilde{p}}_1 \; \ldots\right]$ can each be solved as

$$\left(P^{-1} + C^T R^{-1} C\right) \underline{\tilde{p}}_i = \underline{e}_i, \tag{9.11}$$

where $\underline{e}_i$ is the ith unit vector. However, for an n-element random field $\underline{z}$ there will be n separate linear systems to solve, thus for all but the very smallest problems (9.11) is, at best, an exploratory approach, whereby a few individual columns of $\tilde{P}$ could be computed to acquire some understanding of the behaviour of $\tilde{P}$.

With linear systems solving firmly connected to estimation, the remainder of this chapter surveys seven approaches to solving them, divided into direct and iterative methods. In general, direct methods have the advantage of being exact (excepting numerical errors) and having a fairly predictable computational complexity. The great attraction of iterative methods is their much simpler implementation and their compatibility with sparse A or implicit $\underline{a}()$ of (9.9). However since the iterative methods gradually approach the solution, questions of accuracy and computational complexity may be problem-dependent.

9.1 Direct Solution

Our canonical problem, appearing both in static and dynamic estimators, shown in (9.4), involves a matrix inverse:

$$\hat{\underline{z}}(\underline{m}) = \left(L^T L + C^T R^{-1} C\right)^{-1} C^T R^{-1} \underline{m} \tag{9.12}$$

$$= A^{-1} \underline{b}. \tag{9.13}$$

It may be tempting, given such a problem, to compute the *explicit* matrix inverse A^{-1}, however this is inadvisable on grounds of numerical accuracy. A number of iterative approaches to solving (9.13) are discussed in Section 9.2, however there do exist non-iterative methods, superior to explicit matrix inversion, for solving such systems. These are discussed in the remainder of this section.

9.1.1 Gaussian Elimination

Gaussian Elimination, discussed in Appendix A.7.3, is a direct non-iterative approach to solving linear systems. We apply repeated row operations to the system to convert

$$A\hat{\underline{z}} = \underline{b} \quad \xrightarrow{\text{Row operations}} \quad I\hat{\underline{z}} = \underline{b}'. \tag{9.14}$$

Gaussian elimination is impractical for all but the smallest linear systems for the key reason that in order to be able to apply row operations to A, we need a full, dense, explicit version of A. Furthermore, as Gaussian elimination proceeds, the row operations make A increasingly dense, even if initially sparse. Therefore Gaussian elimination is either inapplicable, or extremely undesirable, given sparse, kernel, or implicit representations of A.

It is common to use the LU decomposition (Appendix A.7.3), which yields triangular matrices L and U, allowing the linear system to be solved by backsubstitution.

9.1.2 Cholesky Decomposition

Because the normal equations in A are known to be symmetric, positive-definite, the Cholesky decomposition is preferred over the LU decomposition for reasons of implementation simplicity, numerical robustness, and computational complexity. The Cholesky decomposition, discussed in Appendix A.7.3, takes a given positive-definite matrix A and computes its matrix square root

$$A = GG^T, \tag{9.15}$$

where G is lower triangular. The method, shown in Algorithm 4, is attractive because of its implementation simplicity (the reader may observe the absence of pivoting

Algorithm 4 The Cholesky Decomposition(many variations exist, see [132, 264])

Goals: Find the matrix square root G of symmetric, positive definite $n \times n$ matrix A

Function $G = \textbf{Chol}(A)$

```
G ← A
for i ← 1 : n do
  for j ← 1 : (i − 1) do
    for k ← i : n do
      g_ki ← g_ki − g_kj · g_ij
    end for
  end for
  g_ii = √g_ii
  for j ← (i + 1) : n do
    g_ji ← g_ji / g_ii
  end for
end for
```

issues common in Gaussian elimination), numerical robustness, and a computational complexity of one half of that of the LU decomposition.

It is the triangularity of G (in both the Cholesky and LU decompositions) which is key to the method's efficiency. Given

$$GG^T \underline{\hat{z}} = \underline{b}, \tag{9.16}$$

using backsubstitution we can easily solve for $\underline{x}$ in

$$G\underline{x} = \underline{b}, \tag{9.17}$$

and then again using backsubstitution to solve for $\underline{\hat{z}}$ in

$$G^T \underline{\hat{z}} = \underline{x}. \tag{9.18}$$

For problems of modest size, solving a linear system using a Cholesky decomposition followed by backsubstitution is a very credible and numerically robust alternative to direct matrix inversion.

9.1.3 Nested Dissection

For problems of large size the Cholesky approach may remain tractable, however subtleties appear. The decomposition algorithm of Algorithm 4 is written in dense form, however for most sparse A the decomposed triangular square root G may or may not be sparse. The degree to which G is denser than A is referred to as *fill-in*, and may be a function of the ordering of rows and columns in A [132], for example whether the state elements in Z should be lexicographically stacked in columns, in rows, or in some other ordering.

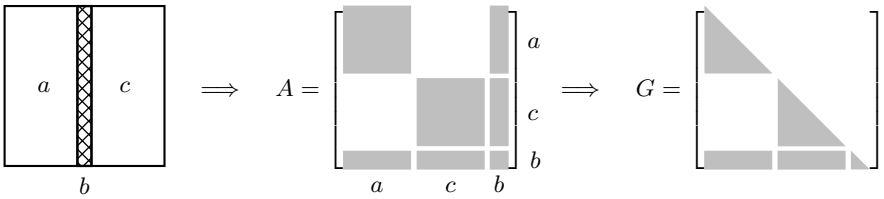


Fig. 9.1. Dividing a domain into decoupled parts, with the boundary elements placed at the end of Z, leads to a reordered A and block-sparse square root G, as shown.

For example, given a small linear system consisting of four elements, the Cholesky factor G ends up dense, even though the linear system possessed sparsity:

$$\begin{bmatrix} \blacksquare & \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & 0 & 0 \\ \blacksquare & 0 & \blacksquare & 0 \\ \blacksquare & 0 & 0 & \blacksquare \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} \quad \Longrightarrow \quad G = \begin{bmatrix} \blacksquare & 0 & 0 & 0 \\ \blacksquare & \blacksquare & 0 & 0 \\ \blacksquare & \blacksquare & \blacksquare & 0 \\ \blacksquare & \blacksquare & \blacksquare & \blacksquare \end{bmatrix} \tag{9.19}$$

In this particular case, if the equations are written in reverse order, the recursive, sequential dependence of the Cholesky method preserves sparsity:

$$\begin{bmatrix} \blacksquare & 0 & 0 & \blacksquare \\ 0 & \blacksquare & 0 & \blacksquare \\ 0 & 0 & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare & \blacksquare \end{bmatrix} \begin{bmatrix} z_4 \\ z_3 \\ z_2 \\ z_1 \end{bmatrix} = \begin{bmatrix} b_4 \\ b_3 \\ b_2 \\ b_1 \end{bmatrix} \quad \Longrightarrow \quad G = \begin{bmatrix} \blacksquare & 0 & 0 & 0 \\ 0 & \blacksquare & 0 & 0 \\ 0 & 0 & \blacksquare & 0 \\ \blacksquare & \blacksquare & \blacksquare & \blacksquare \end{bmatrix} \tag{9.20}$$

For arbitrary nonregular grids or nonstationary MRF kernels, the choice of optimum element ordering to maximize Choleksy sparsity is very difficult and beyond the scope of this text [132, 133, 211]. However, for regular multidimensional grids, Nested Dissection [130, 131] is a method of reordering the elements in a linear system to preserve, as much as possible, the sparsity of the original.

The method relies very much on the Markov principles of conditional decorrelation from Chapter 6. Suppose we can find a boundary which separates the domain into decoupled parts. If $\underline{z}$ is reordered, placing the boundary elements at the *end*, then G preserves the block-sparsity of the decoupled parts, as illustrated in Figure 9.1. The width of the boundary follows immediately from the order of the Markov prior present in the estimation problem leading to A.

Clearly, having inserted one boundary to divide the domain, there is no reason why the domain cannot be further subdivided, leading to the recursive sequence illustrated in Figure 9.2, with obvious generalizations to higher dimensions.

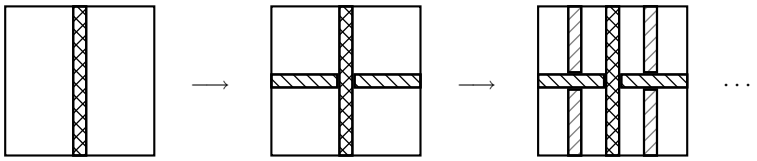


Fig. 9.2. For a two-dimensional problem, further sparsity is realized by repeatedly subdividing along middle columns and rows. The thickness of each boundary portion is a function of the size of the kernel in A, equivalently the order of the Markov prior.

The multiscale approach, discussed in Section 10.3, takes an approach philosophically similar to nested dissection, running a Kalman filter along the sequence of dissection boundaries.

9.2 Iterative Solution

The previous sections considered direct methods to solve a given linear system. However, most direct methods suffer from two difficulties:

1. Implementation complexity, and
2. Computational and storage complexity.

The remainder of this chapter examines iterative methods [153, 278, 346]. We will denote by $\underline{\hat{z}}(k)$ the solution of the linear system solver after k iterations, and by $\underline{e}(k) = \underline{\hat{z}}(k) - \underline{z}$ the error in $\underline{\hat{z}}(k)$, relative to the exact solution $\underline{z}$. If the linear system is well-posed, then the exact solution is guaranteed to exist and to be unique.

Related to the error is the residual $\underline{r}(k) = \underline{b} - A\underline{\hat{z}}(k)$, which measures the degree of inconsistency of some estimate $\underline{\hat{z}}$ with respect to the linear system. We note that

$$\underline{r} = \underline{b} - A\underline{\hat{z}} = \underline{b} - A(\underline{e} + \underline{\bar{z}}) = \underline{b} - A\underline{e} - \underline{b} = -A\underline{e}, \tag{9.21}$$

thus the residual $\underline{r}$ is a transformed version of the error: $\underline{e}$ measures the error in the space of $\underline{z}$; r measures the error in the space of $\underline{b}$.

Where appropriate, $\underline{\delta}(k)$ represents the direction in which we move $\underline{\hat{z}}$ to try to find a better solution. That is, in general,

$$\underline{\hat{z}}(k+1) = \underline{\hat{z}}(k) + \alpha_k \underline{\delta}(k). \tag{9.22}$$

All of the remaining methods in this chapter are iterative and do not require a dense matrix A. What all of the following iterative methods have in common is that the

only thing they require of A is the ability to compute certain matrix–vector products involving A. In particular, the great attraction of the Conjugate Gradient method in Section 9.2.3 is that it only ever requires the product $A\underline{z}$.

9.2.1 Gauss–Jacobi / Gauss–Seidel

The Gauss–Jacobi and Gauss–Seidel iterations [78, 153, 278] are among the simplest linear system solvers, iteratively updating the scalar elements in $\underline{\hat{z}}$ one at a time. It is useful to understand both the scalar and vector forms of the linear system and associated iterations: the scalar form is more closely connected to the algorithmic implementation, whereas the vector form is more convenient for analysis.

If we let

$$A = A_D + A_N \tag{9.23}$$

represent the decomposition of matrix A into diagonal and off-diagonal components, respectively, then we can rearrange the linear system

$$\begin{array}{cc} A\underline{\hat{z}} = \underline{b} & \sum_j a_{i,j}\hat{z}_j = b_i \\ A_D\underline{\hat{z}} + A_N\underline{\hat{z}} = \underline{b} & \left(a_{i,i}\hat{z}_i + \sum_{j\neq i} a_{i,j}\hat{z}_j\right) = b_i \end{array} \tag{9.24}$$

to arrive at the Gauss–Jacobi iteration

$$\underbrace{\underline{\hat{z}}(k+1) = A_D^{-1}\left(\underline{b} - A_N\underline{\hat{z}}(k)\right)}_{\text{Matrix–Vector Form}} \quad \underbrace{\hat{z}_i(k+1) = \frac{b_i - \sum_{j\neq i} a_{i,j}\hat{z}_j(k)}{a_{i,i}}}_{\text{Scalar–Algorithm Form}} \tag{9.25}$$

Similarly, if

$$A = A_L + A_D + A_U \tag{9.26}$$

represents a decomposition into lower-triangular, diagonal, and upper-triangular pieces, then we can derive the Gauss–Seidel matrix–vector form

$$\begin{array}{c} A\underline{\hat{z}} = \underline{b} \\ A_L\underline{\hat{z}} + A_D\underline{\hat{z}} + A_U\underline{\hat{z}} = \underline{b} \\ \underline{\hat{z}}(k+1) = (A_L + A_D)^{-1}\left(\underline{b} - A_U\underline{\hat{z}}(k)\right) \end{array} \tag{9.27}$$

and the scalar–algorithm form:

$$\begin{array}{c} \sum_j a_{i,j}\hat{z}_j = b_i \\ \left(\sum_{j<i} a_{i,j}\hat{z}_j + a_{i,i}\hat{z}_i + \sum_{j>i} a_{i,j}\hat{z}_j\right) = b_i \\ \hat{z}_i(k+1) = \dfrac{b_i - \sum_{j<i} a_{i,j}\hat{z}_j(k+1) - \sum_{j>i} a_{i,j}\hat{z}_j(k)}{a_{i,i}} \end{array} \tag{9.28}$$

Algorithm 5 Gauss–Seidel

Goals: Iteratively solve the linear system $A\underline{x} = \underline{b}$, $\underline{x} \in \mathbb{R}^n$ such that $\|A\underline{\hat{x}} - \underline{b}\| < \zeta$
Function $\underline{\hat{x}} = \mathbf{GS}(A, \underline{b}, \underline{\hat{x}}(0), \zeta)$
  $k \leftarrow 1$
  repeat
    for $i \leftarrow 1 : n$ **do**
      $\underline{\hat{x}}_i(k) \leftarrow \left(b_i - \sum_{j<i} a_{i,j}\hat{x}_j(k) - \sum_{j>i} a_{i,j}\hat{x}_i(k-1)\right)/a_{i,i}$
    end for
    $r_k \leftarrow 0$
    for $i \leftarrow 1 : n$ **do**
      $r_k \leftarrow r_k + \left(b_i - \sum_j a_{i,j}\hat{x}_j(k)\right)^2$ *Compute Residual*
    end for
  until $r_k < \zeta$

The main difference between the Gauss–Jacobi and Gauss–Seidel iterations is that the Gauss–Seidel iteration is *in place*, meaning that only one copy of $\underline{\hat{z}}$ is needed, whereas Gauss–Jacobi needs to preserve all of $\underline{\hat{z}}(k)$ while computing $\underline{\hat{z}}(k+1)$, necessitating two copies.

The simplicity of these two methods allows a straightforward analysis of their convergence properties:

$$\begin{aligned}\underline{e}(k+1) = \underline{\hat{z}}(k+1) - \underline{z} &= A_D^{-1}\big(\underline{b} - A_N\underline{\hat{z}}(k)\big) - \underline{z} \\ &= A_D^{-1}\big(A_D + A_N\big)\underline{z} - A_D^{-1}A_N\underline{\hat{z}}(k) - \underline{z} \\ &= \underline{z} + A_D^{-1}A_N\underline{z} - A_D^{-1}A_N\underline{\hat{z}}(k) - \underline{z} \\ &= -A_D^{-1}A_N\big(\underline{\hat{z}}(k) - \underline{z}\big).\end{aligned} \tag{9.29}$$

That is, the Gauss–Jacobi *errors* obey a dynamic relationship

$$\underline{e}_J(k+1) = -A_D^{-1}A_N\underline{e}_J(k) = -A_D^{-1}\big(A_L + A_U\big)\underline{e}_J(k). \tag{9.30}$$

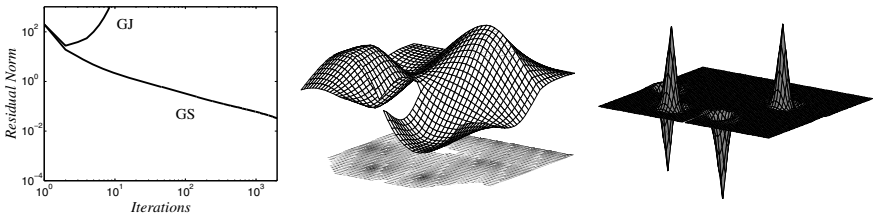
Comparing the Gauss–Jacobi and Gauss–Seidel forms in (9.25), (9.27), the corresponding error dynamics for Gauss–Seidel follow by inspection:

$$\underline{e}_S(k+1) = -\big(A_D + A_L\big)^{-1}A_U\underline{e}_S(k). \tag{9.31}$$

Thus the iterations converge if and only if the eigenvalues of $A_D^{-1}(A_L + A_U)$ or of $(A_D + A_L)^{-1}A_U$, respectively, have a magnitude less than one (meaning that the spectral radius ρ is less than one). Because the spectral radius describes the speed of convergence of the slowest error mode

$$\lim_{k\to\infty}\left(\frac{\|\underline{e}_{k+1}\|}{\|\underline{e}_k\|}\right) = \rho, \tag{9.32}$$

we frequently define a *convergence rate*



Residuals | Exact Solution | Gauss–Seidel Estimate (20 Iterations)

Fig. 9.3. Gauss–Jacobi and Gauss–Seidel residuals, as a function of iteration, for the complex two-dimensional problem of Example 5.1 on page 159.

$$\tau = -\ln(\rho), \tag{9.33}$$

where τ measures the number of iterations for the error to be reduced by a factor of $1/e$, about 60%.

It can be shown [78, 278] that the Gauss–Jacobi and Gauss–Seidel convergence condition is guaranteed if A is strictly diagonally dominant:

$$|a_{i,i}| > \sum_{j \neq i} |a_{i,j}|. \tag{9.34}$$

The corresponding condition on the kernel $\mathcal{A}$ is that A is strictly diagonally dominant if

$$|\mathcal{A}_{0,0}| > \sum_{i,j \neq 0,0} |\mathcal{A}_{i,j}|. \tag{9.35}$$

Thus the membrane kernel of Figure 5.14 on page 158 is strictly diagonally dominant for all $\alpha > 0$, whereas the thin-plate kernel satisfies dominance only if the central element is made rather large, $\alpha > \sqrt{24}$.

In practice the Gauss–Seidel method is typically more robust, converging for a wider variety of problems, and roughly twice as fast as Gauss–Jacobi.

An example of the application of Gauss–Jacobi and Gauss–Seidel to a complex two-dimensional problem is shown in Figure 9.3. Gauss–Jacobi fails to converge, and indeed diverges rapidly. Gauss–Seidel converges, however it is clear from the plotted surface that the estimates respond locally to the measurements with an exponentially-decaying response away from the measurements, precisely along the lines of Figures 8.1 and 8.2.

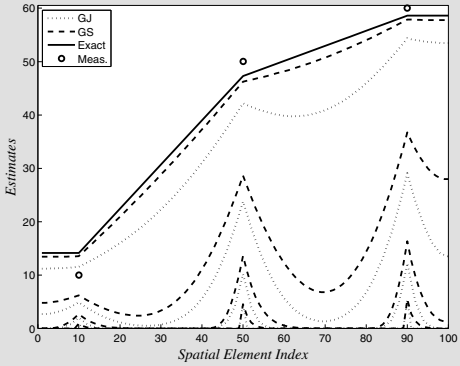
Both Gauss–Jacobi and Gauss–Seidel are very primitive methods and are, on their own, an unrealistic choice for any linear system of significant size, so good performance is in no way expected in Figure 9.3. Modest improvements in computational

Example 9.1: Iterative Solutions to Interpolation

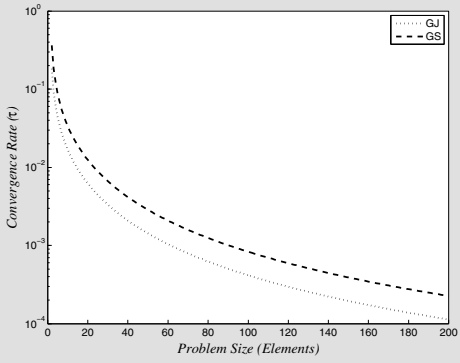
We can consider solving an estimation problem using Gauss–Jacobi or Gauss–Seidel. Suppose we have an n-element one-dimensional random process with first-order regularization constraints, where we measure the 10th, 50th, and 90th elements with unit-variance noise. Thus

$$A = C^T R^{-1} C + \lambda L_x^T L_x, \quad C \in \mathbb{R}^{3\times n} \quad R = I_3 \quad L_x \in \mathbb{R}^{n-1\times n} \tag{9.36}$$

with each row of L_x a first-order penalty term $\begin{bmatrix} 0 \ldots 0 & -1 & 1 & 0 \ldots 0 \end{bmatrix}$, as in Section 5.5. Applying Gauss–Jacobi (9.25) (dotted) and Gauss–Seidel (9.27) (dashed) to this linear system yields



with solutions plotted after 1, 10, 100, 1000 iterations, clearly showing a convergence to the exact solution, plotted as a solid line. The more rapid convergence of the Gauss–Seidel method is clear. It is also clear that convergence is more rapid near measurements. For a given set of measurements the convergence rate τ (9.33) decreases as the process length increases:



Here τ was computed from the exact eigendecomposition of the error dynamics as a function of problem size n.

complexity can be realized by the SOR method (Section 9.2.2). However, for problems of any significant size, to be effective GJ/GS require some sort of problem transformation to break the indirection problem of Figure 8.1, seen vividly in Figure 9.3. A variety of such transformations was discussed in Chapter 8 and preconditioning, specifically of iterative algorithms, is covered in Section 9.2.4.

9.2.2 Successive Overrelaxation (SOR)

The principle of overrelaxation [78,153,278,289] is to take an iterative method, most commonly Gauss–Seidel, and to adjust the change $\big(\underline{\hat{z}}(k+1) - \underline{\hat{z}}(k)\big)$ to accelerate convergence.

Specifically, a given iterative scheme is modified as follows:

$$\begin{array}{ll} \text{Given Iteration:} & \underline{\hat{z}}(k+1) = \underline{\hat{z}}(k) + \underline{\delta}(k) \\ \text{Overrelaxed Form:} & \underline{\hat{z}}(k+1) = \underline{\hat{z}}(k) + \omega\underline{\delta}(k) \quad 0 < \omega < 2, \end{array} \tag{9.37}$$

multiplying the intended adjustment $\underline{\delta}(k)$ by some constant ω.

Computing $\underline{\delta}(k) = \big(\underline{\hat{z}}(k+1) - \underline{\hat{z}}(k)\big)$ as the iteration-to-iteration difference, the overrelaxed forms of the Gauss–Jacobi and Gauss–Seidel iterations can be derived from (9.25), (9.27) as

$$\begin{aligned} \text{GJ: } \underline{\hat{z}}(k+1) &= \underline{\hat{z}}(k) + \omega \underbrace{\left[A_D^{-1}\big(\underline{b} - A_N\underline{\hat{z}}(k)\big) - \underline{\hat{z}}(k)\right]}_{\underline{\delta}(k)} \\ \text{GS: } \underline{\hat{z}}(k+1) &= \underline{\hat{z}}(k) + \omega \underbrace{\left[\left(A_L + A_D\right)^{-1}\big(\underline{b} - A_U\underline{\hat{z}}(k)\big) - \underline{\hat{z}}(k)\right]}_{\underline{\delta}(k)}. \end{aligned} \tag{9.38}$$

The idea is to select ω to reduce the spectral radius of the iteration, thereby accelerating convergence.

Given iterative error dynamics

$$\underline{e}(k+1) = Q\underline{e}(k), \tag{9.39}$$

the effect of ω on the overrelaxed iteration is easily derived:

$$\begin{array}{cc} \text{Basic Iteration} & \text{Overrelaxed} \\ \underline{\hat{z}}(k+1) = \underline{\hat{z}}(k) + \underline{\delta}(k) & \underline{\hat{z}}^{\text{SOR}}(k+1) = \underline{\hat{z}}^{\text{SOR}}(k) + \omega\underline{\delta}(k) \\ \underline{e}(k+1) = \underline{e}(k) - \underline{\delta}(k) \quad \Rightarrow & \underline{e}^{\text{SOR}}(k+1) = \underline{e}^{\text{SOR}}(k) - \omega\underline{\delta}(k) \\ \underline{\delta}(k) = (I-Q)\underline{e}(k) & = \big(I - \omega(I-Q)\big)\underline{e}^{\text{SOR}}(k) \end{array} \tag{9.40}$$

That is, the overrelaxed error dynamics are given by

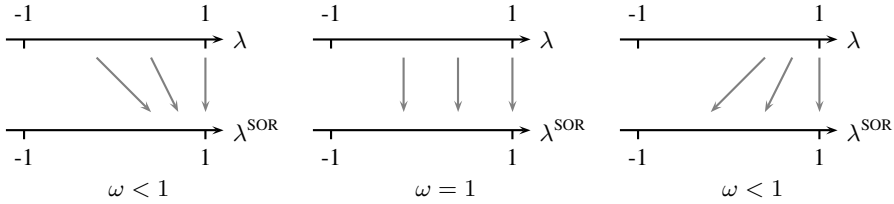


Fig. 9.4. The principle of overrelaxation: Parameter ω controls the degree to which the eigenvalues of the iteration error dynamics are pulled towards 1.0 (left) or pushed away from 1.0 (right). The goal is to choose ω to maximize the distance from ± 1 to the closest eigenvalues.

$$Q^{\text{SOR}}(\omega) = \big(I - \omega(I - Q)\big). \tag{9.41}$$

The key question, then, is how the spectral radii $\rho(Q), \rho\big(Q^{\text{SOR}}(\omega)\big)$ compare. Given the eigendecomposition of Q,

$$Q\underline{v}_i = \lambda_i \underline{v}_i, \tag{9.42}$$

it is possible to analytically derive the eigendecomposition of $Q^{\text{SOR}}(\omega)$:

$$Q^{\text{SOR}}(\omega)\underline{v}_i = \big(I - \omega(I - Q)\big)\underline{v}_i = \big(1 - \omega + \omega\lambda_i\big)\underline{v}_i. \tag{9.43}$$

That is, the eigenvectors of the overrelaxed iteration are unchanged, and the eigenvalues of the error dynamics are modified as

$$\lambda_i \quad \xrightarrow{\text{SOR}(\omega)} \quad \lambda_i^{\text{SOR}}(\omega) = 1 - \omega + \omega\lambda_i, \tag{9.44}$$

So values $0 < \omega < 1$ pull eigenvalues towards 1.0, whereas values $1 < \omega < 2$ push eigenvalues away from 1.0, as illustrated in Figure 9.4. This is actually quite intuitive:

- An eigenvalue less than zero implies an error which oscillates in sign. Therefore we conclude that the iterative step went *too far*, and a smaller step ($\omega < 1$) should push the error eigenvalue closer to zero.
- An eigenvalue close to one implies an error which is decaying only very slowly. Therefore we conclude that the iterative step was *too small*, and a larger step ($\omega > 1$) should help to accelerate the method by pushing the eigenvalue towards zero.

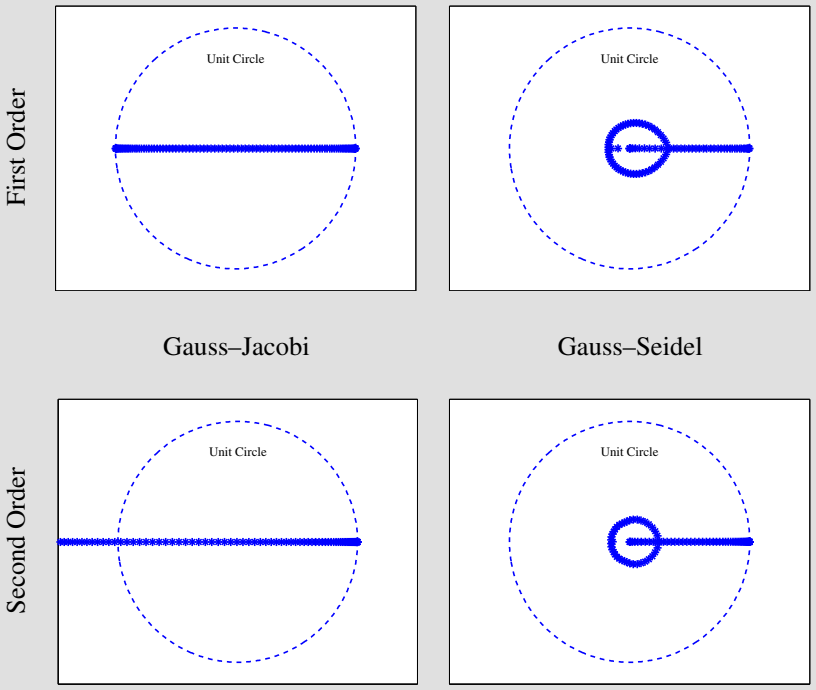
The optimum choice of ω then corresponds to minimizing the spectral radius:

$$\omega_{\text{Optimum}} = \arg_\omega \min \rho\big(Q^{\text{SOR}}(\omega)\big). \tag{9.45}$$

There are two primary limitations to the use of SOR in solving estimation problems:

Example 9.2: Eigendistributions of Iterative Interpolators

Consider the one-dimensional estimation problem of Example 9.1 with a process length of $n = 100$. We consider both first-order (L_x) and second-order constraints (L_{xx}) in (9.36). Four sets of eigenvalues are shown below, plotted in the complex plane:



The instability of Gauss–Jacobi applied to second-order constraints can be seen from the presence of eigenvalues outside of the dashed unit circle.

The eigenvalues of first-order Gauss–Jacobi are symmetric about zero, so SOR has nothing to offer here. However, the Gauss–Seidel eigenvalues are concentrated to one side, so a limited benefit can be gained by overrelaxing Gauss–Seidel with $\omega > 1$, pushing the eigenvalues away from 1.0.

1. A, and therefore Q, are typically huge matrices whose eigendecomposition is unknown. There are special cases for which the optimum ω is known exactly, however, in general finding suitable ω may be a matter of trial and error.

2. If the error dynamics have eigenvalues near one (see Example 9.2) then SOR can, at best, only double the convergence rate τ:

$$\begin{array}{ccc} \lambda = 1 - \epsilon & \xrightarrow{\text{SOR}(\omega=2)} & \lambda^{\text{SOR}} = 1 - 2 + 2\lambda = 1 - 2\epsilon \\ \Downarrow & & \Downarrow \\ \tau = -\ln(1-\epsilon) \simeq \epsilon & & \tau^{\text{SOR}} \simeq 2\epsilon \end{array} \tag{9.46}$$

If there are eigenvalues present with values less than zero, then to prevent instability ω must be chosen somewhat less than two, with a corresponding reduction in convergence rate.

In the limiting case where multiple eigenvalues are distributed symmetrically about zero, then $\omega_{\text{Optimum}} = 1$, implying that SOR offers no benefit.

9.2.3 Conjugate Gradient and Krylov Methods

Conjugate Gradient (CG) is probably the most popular general-purpose method for iterative linear systems solving, finding a nice balance between the trivial implementation and moderate performance of Gauss–Jacobi and Gauss–Seidel, and far more complex methods such as multigrid (Section 9.2.5).

CG is discussed in an enormous number of books and papers [34, 43, 141, 142, 153, 268], however the tutorial in [285] is highly recommended for anyone seeking to understand CG at a deeper, intuitive level.

CG is essentially a method of steepest or gradient descent — sliding down the gradient of some objective function, seeking a minimum. Given the linear system $A\underline{z} = \underline{b}$, where A is symmetric and positive-definite, we define the objective function

$$f(\underline{z}) = \frac{1}{2}\underline{z}^T A\,\underline{z} - \underline{b}^T\underline{z} \tag{9.47}$$

from which it follows that

$$f'(\underline{z}) = \frac{\partial f}{\partial \underline{z}} = A\underline{z} - \underline{b} \qquad f''(\underline{z}) = \frac{\partial^2 f}{\partial \underline{z}^2} = A > \mathbf{0}. \tag{9.48}$$

We see that the first derivative contains our linear system of interest, so our goal is to find $\hat{\underline{z}}$ such that $f'(\hat{\underline{z}}) = 0$, meaning that we have found a minimum of f.

The method of steepest descent seeks to move down the gradient of f to find a minimum. This gradient is found particularly easily:

$$f'\big(\hat{\underline{z}}(k)\big) = A\hat{\underline{z}}(k) - \underline{b} = -\underline{r}(k). \tag{9.49}$$

That is, the residual points in the direction of steepest gradient descent, and the solution is iterated as

$$\hat{\underline{z}}(k+1) = \hat{\underline{z}}(k) + \alpha\underline{r}(k). \tag{9.50}$$

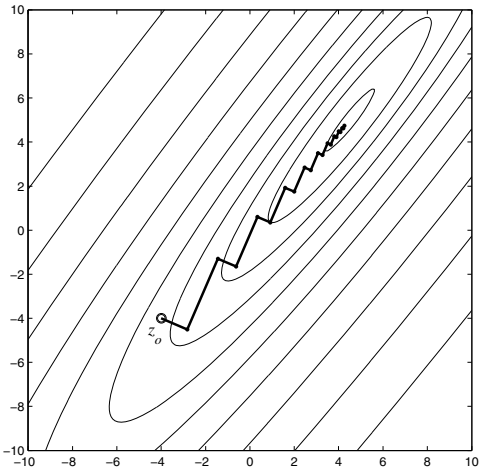


Fig. 9.5. An example of steepest descent. The contours show the values of $f(\underline{z})$, with twenty iterations of steepest descent superimposed. We observe how successive steps move at right angles to each other; it is also clear that many, many steps are taken in the same direction until the desired minimum is reached.

The distance α that we move along $\underline{r}(k)$ is selected to minimize $f\big(\hat{\underline{z}}(k) + \alpha \underline{r}(k)\big)$, thus setting the partial derivative to zero:

$$\begin{aligned} \frac{\partial f\big(\hat{\underline{z}}(k+1)\big)}{\partial \alpha} = 0 \;&\Rightarrow\; f'\big(\hat{\underline{z}}(k+1)\big)^T \frac{\partial \hat{\underline{z}}(k+1)}{\partial \alpha} = 0 \\ &\Rightarrow\; f'\big(\hat{\underline{z}}(k+1)\big)^T \underline{r}(k) = 0 \\ &\Rightarrow\; -\underline{r}(k+1)^T \underline{r}(k) = 0. \end{aligned} \tag{9.51}$$

Substituting $\underline{r}(k+1) = \underline{b} - A\big(\hat{\underline{z}}(k) + \alpha \underline{r}(k)\big)$ allows us to solve for α:

$$\alpha = \frac{\underline{r}(k)^T \underline{r}(k)}{\underline{r}(k)^T A\, \underline{r}(k)}. \tag{9.52}$$

The problem with steepest descent, especially if A is poorly conditioned, is that we may end up converging very slowly, requiring many, many iterations to reach a reasonable answer.

The problem, as is made clear in Figure 9.5, is that steepest descent may end up moving in the same direction many times. Ideally we would go the "right" distance along some direction $\underline{r}(k)$, not such that we end up at a minimum of f (what we did before), which led to the orthogonality

$$\underline{r}(k+1)^T \underline{r}(k) = 0, \tag{9.53}$$

but rather that we move as far in the current direction as we *need* to, implying that the remaining error be orthogonal to the current direction:

$$\underline{e}(k+1)^T \underline{r}(k) = 0. \tag{9.54}$$

Unfortunately we can't do this, since we would need to know the exact solution $\underline{z}$ in order to know $\underline{e}$.

The beauty of conjugate gradient is that it cleverly selects a special set of directions in which to move, which allows steepest descent to move the "right" distance along each direction, eliminating the error along that direction in one iteration. Specifically, if we can find a set of A-orthogonal directions $\{\underline{\delta}(k)\}$, meaning that

$$\underline{\delta}(k)^T A \, \underline{\delta}(j) = 0, \quad j \neq k \tag{9.55}$$

then steepest descent will ensure, after each step, that the error is A-orthogonal to the chosen direction:

$$\underline{e}(k+1)^T A \, \underline{\delta}(k) = 0. \tag{9.56}$$

Having eliminated the error in some direction, it remains eliminated in future iterations. That is, let us suppose that

$$\underline{e}(k)^T A \, \underline{\delta}(m) = 0 \qquad m = 1, \ldots, n-1. \tag{9.57}$$

Then the form of the iteration allows us to see how the error evolves:

$$\underline{\hat{z}}(k+1) = \underline{\hat{z}}(k) + \alpha(k)\underline{\delta}(k) \quad \Rightarrow \quad \underline{e}(k+1) = \underline{e}(k) + \alpha(k)\underline{\delta}(k) \tag{9.58}$$

from which the A-orthogonality of $\underline{e}(k+1)$ follows:

$$\underline{e}(k+1)^T A \, \underline{\delta}(j) = \begin{cases} 0 & j = k \text{ by (9.56)} \\ \underbrace{\underline{e}(k)^T A \, \underline{\delta}(j)}_{=0 \text{ by (9.57)}} + \underbrace{\alpha(k)\underline{\delta}(k)^T A \, \underline{\delta}(j)}_{=0 \text{ by (9.55)}} & j < k \end{cases}$$

Therefore each iteration k permanently eliminates the error along direction $\underline{\delta}(k)$. However, if the linear system is well-posed then A must have full rank, in which case the A-orthogonal directions $\{\underline{\delta}(1), \ldots, \underline{\delta}(n)\}$ must span $\mathbb{R}^n$. Therefore the error will equal zero in n iterations! Our enthusiasm for this conclusion should be tempered by two facts:

1. This number of iterations n is determined by the number of unknowns. For a large 1000×1000 2D spatial problem, the number of iterations would be $n = 10^6$, possibly computationally infeasible.

2. Because of numerical rounding, after n iterations the error is, in fact, not likely to equal zero.

Algorithm 6 Conjugate Gradient

Goals: Iteratively solve the linear system $A\underline{x} = \underline{b}$ with tolerance $\|A\underline{\hat{x}} - \underline{b}\| < \zeta$
Function $\underline{\hat{x}} = \mathbf{CG}(A, \underline{b}, \underline{\hat{x}}_o, \zeta)$
$\underline{d}_o \leftarrow \underline{r}_o \leftarrow \underline{b}$
$k \leftarrow 0$
while $\|\underline{r}_k\| > \zeta$ **do**
$\alpha_k \leftarrow \frac{\|\underline{r}_k\|^2}{\underline{d}_k^T A \underline{d}_k}$ *Steepest Descent*
$\underline{\hat{x}}_{k+1} \leftarrow \underline{\hat{x}}_k + \alpha_k \underline{d}_k$
$\underline{r}_{k+1} \leftarrow \underline{r}_k - \alpha_k A \underline{d}_k$ *Update Residual*
$b_k \leftarrow \frac{\|\underline{r}_{k+1}\|^2}{\|\underline{r}_k\|^2}$ *Gram–Schmidt for Next Direction*
$\underline{d}_{k+1} \leftarrow \underline{r}_{k+1} + b_k \underline{d}_k$
$k \leftarrow k + 1$
end while

The key question is how to find the A-orthogonal directions $\{\underline{\delta}(k)\}$. Given a set of linearly independent vectors $\{\underline{u}(k)\}$, the conjugate Gram–Schmidt procedure (Appendix A.7.3) can compute an A-orthogonal set. Because the residuals $\{\underline{r}(k)\}$ can be shown to be orthogonal, and therefore linearly independent, they are a straightforward choice: $\underline{u}(k) = \underline{r}(k)$. If we let $\mathcal{D}_k$ represent the space explored after the first k iterations, then by definition

$$\mathcal{D}_n = \text{span}\,\{\underline{\delta}(1), \underline{\delta}(2), \ldots, \underline{\delta}(k)\} \tag{9.59}$$
$$= \text{span}\,\{\underline{r}(1), \underline{r}(2), \ldots, \underline{r}(k)\}\,. \tag{9.60}$$

However, because the residuals obey the recursion

$$\underline{r}(k) = \underline{r}(k-1) - \alpha(k-1) A \underline{\delta}(k-1), \tag{9.61}$$

it follows that the space $\mathcal{D}_k$ equals the space $\mathcal{D}_{k-1}$ extended with direction $A\underline{\delta}(k-1)$. However, as $\underline{\delta}(k-1) \in \mathcal{D}_{k-1}$, it follows that

$$\mathcal{D}_k = \mathcal{D}_{k-1} \oplus A\mathcal{D}_{k-1} \tag{9.62}$$

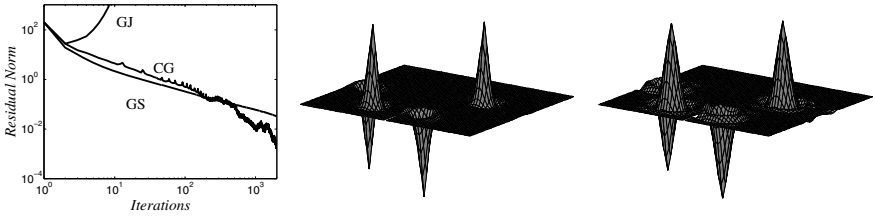
from which it follows that

$$\mathcal{D}_k = \text{span}\left\{\underline{\delta}(1), A\underline{\delta}(1), \ldots, A^{k-1}\underline{\delta}(1)\right\} \tag{9.63}$$
$$= \text{span}\left\{\underline{r}(1), A\underline{r}(1), \ldots, A^{k-1}\underline{r}(1)\right\}. \tag{9.64}$$

This sort of space construction is known as a Krylov space — the repeated application of a linear operator A to some initial vector.

The form of this Krylov space is particularly convenient because the residual $\underline{r}(k)$ is already A-orthogonal to all of $\underline{\delta}(1), \ldots, \underline{\delta}(k-2)$, leaving only one step of Gram–Schmidt to remove the relationship with $\underline{\delta}(k-1)$.

We are thus left with the elegant conjugate-gradient procedure of Algorithm 6, consisting of three iterated steps:



Residuals | Gauss–Seidel Estimate (GS: 20 Iterations) | Conjugate Gradient Estimate (CG: 20 Iterations)

Fig. 9.6. Continuing from Figure 9.3, looking at a complex two-dimensional system, comparing Gauss–Jacobi, Gauss–Seidel, and Conjugate Gradient.

1. Do one step of steepest descent, eliminating the error in the current direction.
2. Update the residual based on the changed value of the estimate.
3. Based on this residual perform one step of conjugate Gram–Schmidt to find the next direction.

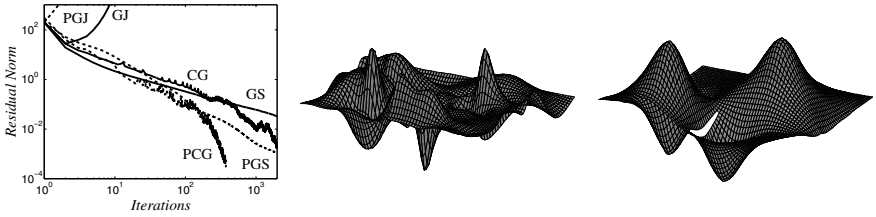
A variety of criteria may be proposed by which the algorithm is stopped, whether one iteration per state element (for the nominally exact answer, excepting numerical errors), a fixed predetermined number of iterations, or based on the size of the residual.

Figure 9.6 shows a two-dimensional estimation problem, comparing the convergence of Gauss–Seidel and conjugate gradient. Although the difference in convergence is not, in this particular example, terribly striking, conjugate gradient has a *major* advantage: whereas Gauss–Jacobi and Gauss–Seidel require computing certain diagonal or off-diagonal elements of A, the *only* thing which conjugate gradient requires is an ability to compute the matrix–vector product $A\underline{z}$. This advantage is significant when A is algorithmically generated, as in (9.9), and particularly in the context of preconditioning, discussed in Section 9.2.4.

Conjugate gradient is only one of many Krylov algorithms [168, 321]. Other widely-used forms include GMRES [277, 278] and biconjugate gradient [278].

9.2.4 Iterative Preconditioning

Preconditioners [39, 63, 79, 98, 348] were discussed in the context of conditioning inverse problems in Section 2.3, and many of the methods discussed in Chapter 8 are essentially preconditioners — changes or reductions of basis in order to make a given problem numerically or computationally simpler.



Residuals | Preconditioned GS (PGS: 20 Iterations) | Preconditioned CG (PCG: 20 Iterations)

Fig. 9.7. This figure shows the preconditioned version of Figure 9.6. Clearly the preconditioned approaches require fewer iterations to converge; however in the case of GS, preconditioning significantly increases the computational complexity *per iteration* because the system matrix A is made less sparse. GS was preconditioned using a DB4 wavelet and CG using the hierarchical triangles of Section 8.4.1.

Assume, then, that after whatever problem reduction / decoupling of Chapter 8 has been applied, that we are left with a linear system to solve. For linear system problems of any significant size, unless the problem is very well conditioned (e.g., an exponential prior model with a short correlation length), it is almost certainly inappropriate to be considering a plain Gauss–Seidel or conjugate-gradient implementation.

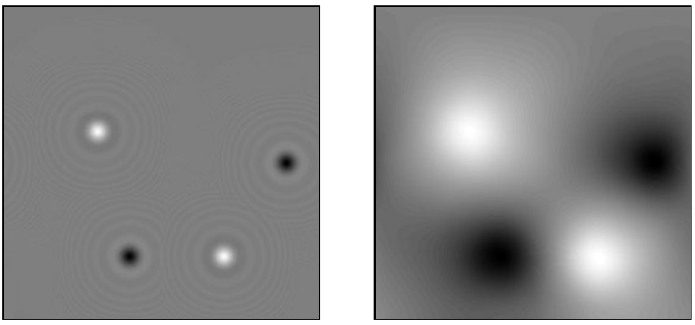
All preconditioners follow the basic approach seen in (8.84) in Chapter 8:

$$\begin{array}{ccccc} A\underline{z} = \underline{b} & \longrightarrow & S^T A\, S\bar{\underline{z}} = S^T\underline{b} & \longrightarrow & \bar{A}\bar{\underline{z}} = \bar{\underline{b}} \\ & & & & \Big\downarrow \text{ Easy ?} \\ \hat{\underline{z}} & \longleftarrow & \text{S} & & \hat{\bar{\underline{z}}} \end{array} \tag{9.65}$$

The two fundamental questions, then, are the possible choices of preconditioner S, and the associated computational issues.

The Gauss–Seidel algorithm considers each state element individually, in turn, so some sort of preconditioner providing state mixing is needed, such as the DB4 orthogonal wavelet preconditioner in Figure 9.7. Because the Gauss–Seidel algorithm makes explicit reference to the elements of A, the preconditioned system matrix $\bar{A} = S^T A\, S$ needs to be calculated and stored *explicitly*. In the 2D problem of Figure 9.7, although preconditioned GS required fewer iterations to converge, the computational complexity *per iteration* was increased by a factor of 58:

$2 \cdot 10^4$ Nonzero Values in A $\qquad$ $1 \cdot 10^6$ Nonzero Values in $\bar{A} = S^T A\, S$



Conjugate Gradient Estimates (100 Iterations)

Preconditioned CG Estimates (100 Iterations)

Fig. 9.8. Conjugate gradient can be applied to large-scale problems, here the cut–fold problem of Example 5.1 and Figure 9.6, but sixteen times larger on a 257×257 domain.

The preconditioning of conjugate gradient is a bit more subtle [285], and the following discussion can only be considered an introduction. Conjugate gradient iterates over directions $\underline{\delta}(k)$, rather than over coordinates as does Gauss–Seidel, therefore rotating the entire problem domain has no effect on CG. Because an orthogonal matrix is essentially a multidimensional rotation, orthogonal preconditioners (such as the Daubechies wavelets) therefore have no effect on CG.

The beauty of CG is that requiring only the matrix–vector product $A\underline{z}$ allows the preconditioning to be implemented implicitly — never does $\bar{A}$ need to be stored. That is, under a preconditioner S, the product $\bar{A}\underline{z}$ is easily computed as

$$\bar{A}\underline{z} = S^T A\, S\underline{z} = S^T\Big(A\big(S\underline{z}\big)\Big). \tag{9.66}$$

Indeed, not even the preconditioner S and the system matrix A need to be stored explicitly. Given our usual constraints matrix L, implying

$$A = C^T R^{-1} C + \lambda L^T L, \tag{9.67}$$

then the product $\bar{A}\underline{z}$ should be found implicitly as

$$\underline{\bar{z}} = \texttt{FunctionS}(\underline{z}) \tag{9.68}$$

$$\bar{A}\underline{z} = \texttt{FunctionST}\big(C^T R^{-1}(C\underline{\bar{z}}) + \lambda L^T(L\underline{\bar{z}})\big), \tag{9.69}$$

where $\texttt{FunctionS}(\underline{x})$ and $\texttt{FunctionST}(\underline{x})$ are algorithms, returning the matrix–vector products $S\underline{x}$, $S^T\underline{x}$, respectively. With such an approach, although $\bar{A}$ may have a higher density, these elements are never explicitly computed or stored, so the preconditioner adds very little computational overhead.

This latter, implicit approach allows preconditioned CG to be applied to much larger linear systems than GS. Figure 9.8 shows conjugate gradient applied to a problem of size 257×257, having $257^2 = 66\,049$ unknowns, which was solved using (9.69) in MATLAB on a regular computer.

9.2.5 Multigrid

The previous sections have looked at very common, general-purpose methods of solving linear systems, methods that appear in countless textbooks and come as standard subroutines in numerical packages such as MATLAB. In this section we want to look at multigrid, a method which focuses explicitly on the solving of large, multidimensional linear estimation problems.

The multigrid method seeks a hierarchical approach to solving a linear system, representing a multidimensional problem at a number of resolutions. We have seen similar hierarchical approaches before:

- Nested dissection (Section 9.1.3) sought to divide a given spatial problem into decoupled pieces, an approach that was particularly useful for random fields governed by a low-order Markov prior.

  However, although nested dissection is hierarchical, it is not multiresolution. That is, the elements being manipulated and rearranged are all individual state elements from the finest scale.

- Hierarchical changes of basis (Section 8.4) converted a linear system on a single scale into a new, changed linear system on a hierarchy of scales, which could then be solved via methods such as GS or CG.

  The hierarchical basis with an iterative solver has a great deal in common with multigrid, however whereas a hierarchical basis leaves us with a single problem, distributed across resolutions, the multigrid method proposes an ensemble of coupled problems, one problem at each resolution, a somewhat simpler mental picture.

The multigrid method [40, 43, 152, 153, 228, 229, 289, 330] is widely used and documented; this section emphasizes the intuition behind the method, but does not give a detailed and comprehensive derivation of the method, and the reader is directed towards the many references.

For precisely the same reasons that a multiresolution change-of-basis was proposed in Section 8.4 — the locality of operator A, the slow spread of information from one state element to another — we now consider the multigrid algorithm, a multiresolution approach to multidimensional linear systems problems. The effect of locality is made clear in Figure 9.9; after very few iterations the portions of the error at high

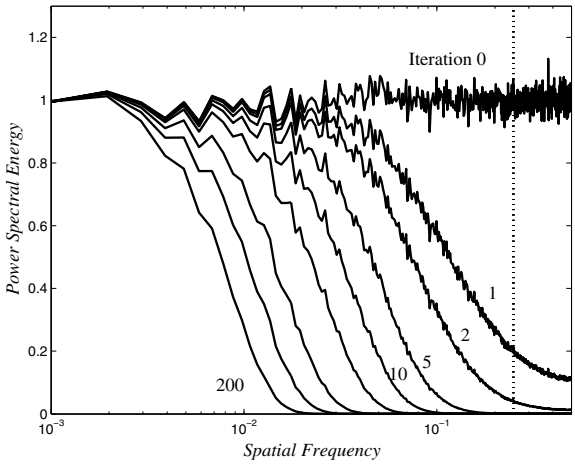


Fig. 9.9. The power spectrum of the estimation error as a function of Gauss–Seidel iterations: After only five iterations the high-frequency error has disappeared, however hundreds or thousands of iterations would be required to reduce the large-scale error.

spatial frequencies have been eliminated, but hundreds or thousands of iterations would be required to significantly reduce long-range errors.

The key idea of multigrid is this:

1. *If only a few iterations of a simple linear system method can eliminate high-frequency errors, then the error quickly becomes smooth.*
2. *However, a smooth (slowly varying) problem can be subsampled, making the remaining long-range errors more local.*
3. *These local errors will then once again reduce relatively rapidly when a linear system method is applied to the subsampled problem.*

Thus, given an initial estimate $\underline{\hat{z}}_o$, after k Gauss–Seidel iterations the new estimate $\underline{\hat{z}}(k)$ has a corresponding residual

$$\underline{r}(k) = \underline{b} - A\underline{\hat{z}}(k) \tag{9.70}$$

such that $\underline{r}(k)$ is spatially smooth. Recall from Section 8.2 that pseudoinverse interpolation S and subsampling F operators will satisfy $FS = I$, but $SF \neq I$. However, the smoothness of $\underline{r}$ means that it can be losslessly subsampled,

$$SF\underline{r}(k) \simeq \underline{r}(k). \tag{9.71}$$

Algorithm 7 Multigrid

Goals: Iteratively solve the linear system $A\underline{x} = \underline{b}$ with tolerance $\|A\underline{\hat{x}} - \underline{b}\| < \zeta$
Function $\underline{\hat{x}} =$ **Multigrid**$(scales, A, \underline{b}, \underline{\hat{x}}_o, \zeta)$
  $\underline{\hat{x}} \leftarrow \underline{\hat{x}}_o$
  repeat
    Call $\underline{\hat{x}} \leftarrow$ **MG**$(scales, A, \underline{b}, \underline{\hat{x}})$
  until $\|A\underline{\hat{x}} - \underline{b}\| < \zeta$

Goals: Perform one complete multigrid cycle, recursively from finest to coarsest scale
Function $\underline{\hat{x}} =$ **MG**$(scale, A, \underline{b}, \underline{\hat{x}}_o)$
  From $\underline{\hat{x}}_o$, perform k iterations of Gauss–Seidel, giving $\underline{\hat{x}}_k$
  if $scale > 0$ **then**
    $\bar{A} \leftarrow S^T A\, S$
    $\underline{\bar{b}} \leftarrow S^T(\underline{b} - A\underline{\hat{x}}_k)$
    $\underline{\bar{x}}_0 \leftarrow \underline{0}$
    for $i \leftarrow 1$ to p **do**
      $\underline{\bar{x}}_i \leftarrow$ **MG**$(scale - 1, \bar{A}, \underline{\bar{b}}, \underline{\bar{x}}_{i-1})$
    end for
    $\underline{\hat{x}}_p \leftarrow \underline{\hat{x}}_k + S\underline{\bar{x}}_p$
    From $\underline{\hat{x}}_p$, perform k iterations of Gauss–Seidel, giving $\underline{\hat{x}}$
  else
    We are at coarsest scale, solve exactly for $\underline{\hat{x}}$.
  end if

The subsampled problem $F\underline{r}(k)$ will converge rapidly at first: it is smaller in size than the original problem, thus each iteration is of reduced complexity, and $F\underline{r}(k)$ will be rougher (less smooth) than $\underline{r}(k)$ and therefore more quickly smoothed by Gauss–Seidel. This concept is illustrated in Figure 9.10 for a one-dimensional problem, repeatedly subsampled three times.

Recall from (9.21) that $e(k)$ represents the error in solution $\underline{\hat{z}}(k)$ after k iterations, where

$$\underline{r}(k) = -A\underline{e}(k). \tag{9.72}$$

For most priors A is a spatial smoothing operator, so we need to oversmooth $\underline{r}$ to ensure that $\underline{e}$ is smooth, allowing it to be estimated

$$\underline{\hat{e}}(k) = S\underline{\hat{z}}_c \tag{9.73}$$

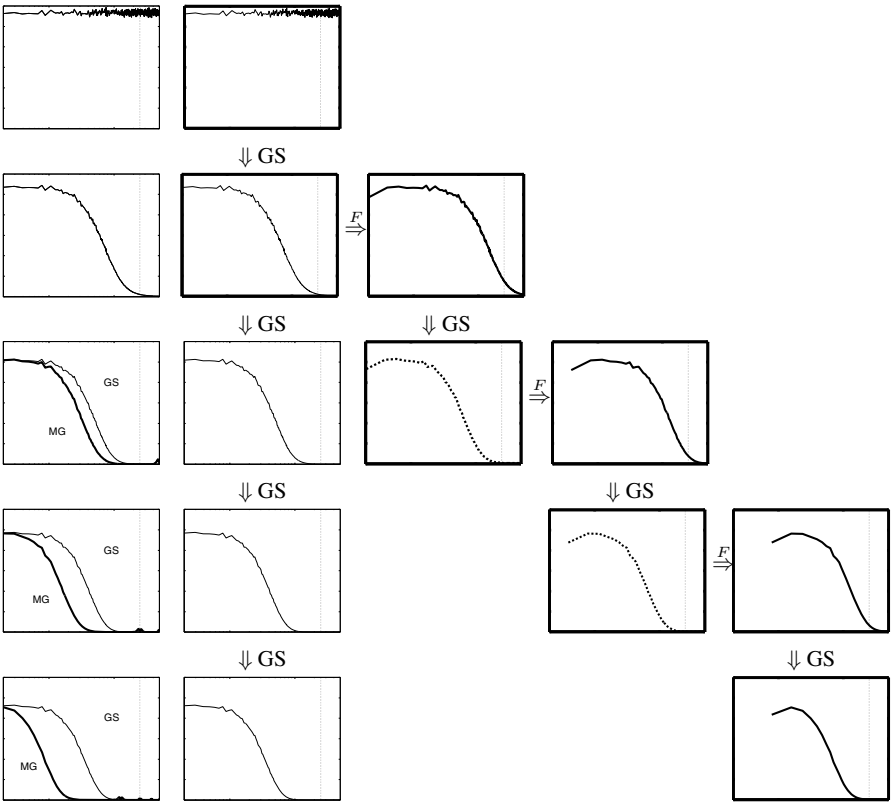
by coarse-scale approximation $\underline{\hat{z}}_c$. The idea, then, is that $\underline{\hat{z}}$ provides the local details, and $S\underline{\hat{z}}_c$ the nonlocal, large-scale parts. $\underline{\hat{z}}_c$ must satisfy

$$A\underline{z} = \underline{b} \tag{9.74}$$

$$A(\underline{\hat{z}} + S\underline{\hat{z}}_c) = \underline{b} \tag{9.75}$$

$$S^T A\, S\underline{\hat{z}}_c = S^T\left(\underline{b} - A\underline{\hat{z}}\right) \tag{9.76}$$

from which we can derive the subsampled linear-system



Comparison At Finest Scale	Finest Scale GS	Multigrid Scale 1	Multigrid Scale 2	Multigrid Scale 3

Fig. 9.10. The multigrid principle: A simple, local iterative method, such as Gauss–Seidel, very quickly eliminates high-frequency (local) errors, but only very slowly eliminates low-frequency (distant) ones. Each panel plots the log power spectrum of the error, as in Figure 9.9. In each row five additional GS steps have been performed relative to the previous row; additional columns are introduced as the system is subsampled (F) to coarser scales. The leftmost column compares GS and multigrid: it is clear that repeated subsampling leads to a much faster reduction in error, as opposed to finest-scale GS with no subsampling. The bolded panels show the sequence followed by the multigrid algorithm.

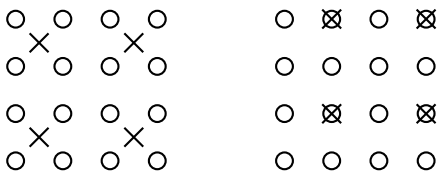


Fig. 9.11. A subsampling operator must be chosen to define the coarser scale. A cell-centred approach, left, is very simple, however the coarse-scale pixels ($\times$) are in a different location from the fine-scale ones ($\circ$), making the assertion of boundary conditions possibly more difficult. In many circumstances the vertex-centred approach, right, is preferred.

$$\underbrace{S^T A S}_{\bar{A}} \underline{\hat{z}}_c = \underbrace{S^T \underline{r}}_{\bar{\underline{b}}} . \tag{9.77}$$

This transformation we have seen before, in (8.7) or (8.84), and is nothing more than an implicit basis reduction over scale.

This new linear system can then, itself, be subsampled after a few iterations of Gauss–Seidel, and the whole smoothing–subsampling process repeats recursively until the original problem has been subsampled to such a degree that it can be solved exactly (e.g., by matrix inversion). The recursive procedure is illustrated in Algorithm 7.

Although the multigrid algorithm may feel rather complex, in fact the actual implementation of the multigrid method is fairly straightforward, as illustrated by the author's MATLAB implementation in Algorithm 8, very nearly line-by-line mirroring the pseudocode in Algorithm 7. The implementation in Algorithm 7 is based on the *explicit* evaluation and storage of matrix A at each scale, however because multigrid only ever requires the matrix–vector product $A\underline{z}$ it is possible, and indeed highly desirable for very large problems, to represent matrix A *implicitly* via a kernel, sparse matrix, or function. A fairly elegant approach for computing a functional, implicit form of A at every scale is shown in Algorithm 9. Because the Gauss–Seidel iterations used in Algorithm 8 require access to a regular, non-functional A, in Algorithm 9 the Gauss–Seidel iterations at each scale have been replaced with conjugate gradient.

There are a number of choices within the multigrid algorithm, the most fundamental of which is the choice of scale-to-scale subsampling operator. The subsampler may be cell-centred or vertex-centred [190], as illustrated in Figure 9.11, the subsampler may be pointwise (frequently leading to an unstable iteration), more commonly a bilinear interpolation, or based on wavelets [44, 316].

Algorithm 8 MATLAB Multigrid implementation I, mirroring Algorithm 7

```
% For convenience, mg is written in terms of subsampler F rather than interpolator S
function x = mg( scale, a, b, x, p, numgs )
if (scale > 0)
    % Do GS iterations
    x = gs(a,b,x,numgs);

    % Make simple 2x2 subsampler F
    f = subsampler(scale);

    % Create coarser problem AC * XC = BC
    ac = sparse(f*a*f');
    bc = f*(b-a*x);
    xc = 0*bc;

    % Repeatedly call coarser scale
    for i=1:p,
        xc = mg( scale-1, ac, bc, xc, cyc, numgs );
    end

    % Add coarse solution back in to current scale and GS iterations to smooth
    x = x + f'*xc;
    x = gs(a,b,x,numgs);
else
    % We are at coarsest scale, solve outright
    x = a \ b;
end
```

A second choice is that of variable p in Algorithm 7, addressing the question of *cycle* — how the algorithm switches between scales. Most common are the "V"-cycle ($p = 1$) the "W" cycle ($p = 2$). As p is increased, a higher proportion of time is spent at the coarser scales; whether this helps will be a function of the problem and needs to be determined empirically.

Other choices include whether to do GS, as in Algorithm 8, or possibly conjugate gradient, as in Algorithm 9, and then whether before, after, or both before and after the coarser scale call. There are also practical considerations, whether the system matrix A and the subsampler S are dense or implicit, and whether the matrix representation may be a function of scale (e.g., implicit at large, fine scales and dense at small, coarse scales).

Figure 9.12 plots the convergence results, comparing multigrid and preconditioned conjugate gradient. The multigrid results were computed using the code in Algorithm 9.

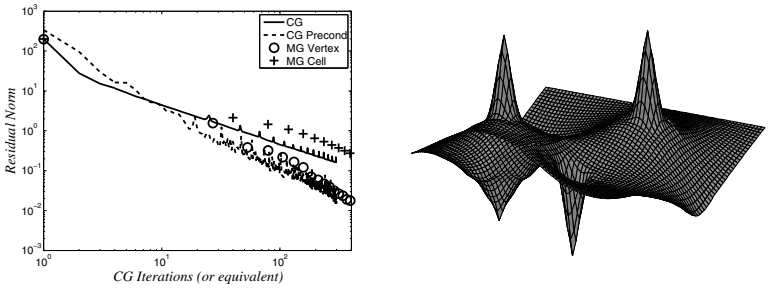
Algorithm 9 MATLAB Multigrid implementation II, using implicit, computed system and scale-transfer matrices

```
% afn is a function handle, such that afn(x) evaluates A*x
% subfn and subfnt are function handles to the subsampler and its transpose
function x = mg( scale, siz, afn, b, subfn, subfnt, x, p, numcg )
if (scale > 0)
    % Do CG iterations
    x = pcg(afn,b,[ ],numcg,[ ],[ ],x);

    % Create a handle to implicit coarser-scale problem
    ac = @(x) subfn(afn(subfnt(x)));
    bc = f*(b-afn(x));
    xc = 0*bc;

    % Repeatedly call coarser scale
    for i=1:p,
        xc = mg( scale-1, ac, bc, subfn, subfnt, xc, cyc, numgs );
    end

    % Add coarse solution back in to current scale and CG iterations to smooth
    x = x + subfnt(xc);
    x = pcg(afn,b,[ ],numcg,[ ],[ ],x);
else
    % We are at coarsest scale, solve well
    x = pcg(afn,b,[ ],100,[ ],[ ],x);
end
```



Residuals for Large Problem MG Estimates (One Iteration)

Fig. 9.12. A comparison of the convergence of Conjugate Gradient and Multigrid. The multigrid estimates were taken after *one* iteration of multigrid, corresponding to twenty iterations of GS or CG (compare with Figure 9.6). The convergence results, left, are based on the 257×257 problem of Figure 9.8, clearly showing the bilinear vertex-centred multigrid outperforming the cell-centred case.

Application 9: Surface Reconstruction

The problem of surface reconstruction [27, 160, 161, 304] is a topic of interest in the field of computer vision, involving the estimation of an unknown surface based on a set of noisy measurements of some function of the surface, possibly of its derivatives, and based on a prior model to regularize the problem.

We have seen repeated examples of surface reconstruction, in the context of membrane and thin-plate models in Chapter 5, and in particular the two-dimensional reconstructions throughout this chapter.

The surface S of interest is a two-dimensional function $s(x, y)$, twice differentiable, with gradients

$$p(x,y) = s_{\mathrm{x}}(x,y) = \frac{\partial s(x,y)}{\partial x} \qquad q(x,y) = s_{\mathrm{y}}(x,y) = \frac{\partial s(x,y)}{\partial y}. \tag{9.78}$$

In terms of choosing a state to represent the surface, we have three alternatives:

1. Estimate the gradients only,

$$z(x,y) = \begin{bmatrix} p(x,y) \\ q(x,y) \end{bmatrix}, \tag{9.79}$$

in which case the estimated surface $\hat{s}$ must be found by integrating $\hat{p}, \hat{q}$. For the integral to be unique (path-independent), p, q must be gradients of a surface, implying a consistency constraint [118, 159]

$$\oint (p dx + q dy) = 0 \tag{9.80}$$

must hold over all closed paths in the plane. Because the surface is not computed until *after* estimation, this formulation does not allow measurements of the surface, and has limited use. There are important exceptions however, such as shape-from-shading [118, 161, 167, 201, 251, 349], which have only gradient measurements.

2. *Jointly* estimate the surface and its gradients,

$$z(x,y) = \begin{bmatrix} s(x,y) \\ p(x,y) \\ q(x,y) \end{bmatrix}, \tag{9.81}$$

in which case it is straightforward to have measurements of either gradients or surface, or to assert prior constraints on the gradients. However, there is the burden of constraining the relationship between s and p, q:

$$\int \int \left(s_{\mathrm{x}} - p\right)^2 + \left(s_{\mathrm{y}} - q\right)^2 dx\, dy. \tag{9.82}$$

3. Estimate the surface directly,

$$z(x,y) = \left[s(x,y)\right], \tag{9.83}$$

in which prior constraints on or measurements of the gradients need to be related to differences of surface values. Although straightforward, this approach may not be preferred in cases where we wish to calculate gradient error statistics, or where only measurements of gradients are available.

In all cases, given linear measurements and prior constraints, the resulting problem reduces to the familiar linear system

$$\left(L^T L + C^T R^{-1} C\right) \underline{\hat{z}}(\underline{m}) = C^T R^{-1} \underline{m}. \tag{9.84}$$

The linear systems methods in this chapter are well represented in the surface reconstruction literature:

- Conjugate gradient, for surface reconstruction using hierarchical interpolants [299],
- Conjugate gradient, for surface reconstruction using wavelet preconditioning [345],
- Multigrid, for surface reconstruction with cuts and folds [303, 304],
- Multiscale, related to nested dissection, for surface estimation with error statistics [108].

For Further Study

There are a great many numerical methods texts, most of which cover the solving of linear systems; one such text is the classic by Dahlquist and Bjorck [78]. The text by Hackbusch [153] parallels the sequence of methods of this chapter, but in far greater detail, and is strongly recommended.

For a more advanced look at sparse methods and solvers, the book by George and Liu [131] is widely cited.

For iterative methods more specifically, the survey paper by Shewchuck [285] is well worth reading to better understand conjugate gradient. The recent text by Saad [278] is comprehensive and up-to-date, giving extensive coverage to GMRES and Krylov methods. Saad's book does not, however, look at multigrid; for accessible treatments of the multigrid method the texts of Briggs [43], Hackbusch [152], McCormick [228], and Wesseling [330] are recommended.

Although the conjugate-gradient method and the Cholesky decomposition are standard components of most numerical packages, such as MATLAB, *Numerical Recipes* [264] remains a valuable resource for readers seeking to do their own implementations, or to develop variations on standard methods.

Sample Problems

Problem 9.1: Numerics of Linear Systems

The solution to a linear system $A\underline{z} = \underline{b}$ can be written algebraically as

$$\underline{\hat{z}} = A^{-1}\underline{b} \tag{9.85}$$

under the assumption that A is square and invertible.

Numerically, however, it was claimed in Section 9.1 that explicitly inverting A is inferior to Gauss / Cholesky methods. In MATLAB, the two alternative implementations to (9.85) would be written as

$$\texttt{Z} = \texttt{inv(A)} * \texttt{b}; \qquad \texttt{Z} = \texttt{A} \setminus \texttt{b}; \tag{9.86}$$

Construct a linear system, for example the one-dimensional interpolation of Problem 9.2, and compare the computational time and accuracy of the two computed estimates in (9.86).

Problem 9.2: Iterative Solutions and Interpolation

We wish to reproduce and extend some of the results of Example 9.2. We have a process of length $n = 100$, of which we measure first and last elements:

$$\begin{bmatrix} 10 \\ 50 \end{bmatrix} = \underline{m} = C\underline{z} + \underline{v} = \begin{bmatrix} z_1 + v_1 \\ z_n + v_2 \end{bmatrix}$$

We propose two possible linear systems to solve:

1. First Order: $A_1 = C^T C + 5 \cdot L_1^T L_1 \quad \longrightarrow \quad A_1\underline{z} = C^T\underline{m}$
2. Second Order: $A_2 = C^T C + 2000 \cdot L_2^T L_2 \quad \longrightarrow \quad A_2\underline{z} = C^T\underline{m}$.

If the iteration is stable, as determined by its eigenvalues, then its time constant is determined by the eigenvalue which is largest in magnitude:

$$\tau = \frac{-1}{\ln\left(\max_i(|\lambda_i|)\right)}$$

(a) Are GJ and GS stable for both the first- and second-order interpolation problems?
(b) For $n = 3, 4, \ldots, 100$ plot the time constant τ for both GJ and GS for the first-order problem. What do you observe?
(c) For $n = 100$, plot all of the eigenvalues in the complex plane. Generate two plots, one for each of GJ and GS, for the first-order problem. What do you observe?
(d) An eigenvalue describes how rapidly an error decays, where the shape of the error is described by the associated eigenvector. Plot two eigenvectors for the first-order GS problem, one corresponding to an eigenvalue near 1.0, another corresponding to an eigenvalue near 0.0. What do you observe?
(e) For $n = 100$, apply GS to the first- and second-order problems, initializing with $\underline{z}_0 = \underline{0}$. What do you observe?

Problem 9.3: Two-Dimensional Interpolation

We wish to solve a two-dimensional estimation problem. Let Z be a 100×100 random field, characterized by constraints

$$L_2 = \begin{bmatrix} L_{xx} \\ L_{yy} \end{bmatrix}$$

with free boundary conditions. Use a regularization constant $\lambda = 5000$.

We have four measurements, assumed to have unit-variance noise:

$$\underline{m} = \begin{bmatrix} \text{-}10 \\ 10 \\ 10 \\ \text{-}10 \end{bmatrix} = \begin{bmatrix} z_{25,25} + v_1 \\ z_{25,75} + v_2 \\ z_{75,25} + v_3 \\ z_{75,75} + v_4 \end{bmatrix}$$

(a) The system matrix A is a large, sparse 10000×10000 array. Do *not* create matrices L_2 or A, not even sparse forms.
Rather, develop a method, an algorithm (e.g., in MATLAB) for computing the two parts of the matrix–vector product

$$A\underline{z} = (C^T C + \lambda L_2^T L_2)\underline{z} = C^T C\underline{z} + \lambda L_2^T L_2 \underline{z}$$

The term $C^T C\underline{z}$ is easily implemented; you will probably want to use a convolution to calculate $L_2^T L_2 \underline{z}$.
(b) Given the matrix–vector product in part (a), you can now implement an iterative solver. Implement CG and plot the results after 10, 100, and 1000 iterations.
(c) Run CG for many iterations, until you feel the method has converged, to acquire $\hat{\underline{z}}_{opt}$. We can calculate the mean-square error at any iteration by comparing the iterative estimate $\hat{\underline{z}}_i$ to the optimum:

$$MSE_i = \|\underline{\hat{z}}_i - \underline{\hat{z}}_t extopt\|.$$

Plot $MSE_i, 1 \leq i \leq 1000$ for CG. Comment.

(d) Modify your algorithm from (a) to yield the needed nonzero elements of A in order to implement GS. Plot MSE_i, comparing GS and CG.

Problem 9.4: Multigrid

Problem 9.3 set up a two-dimensional estimation problem. Solve the estimation problem for $\hat{Z}$ using multigrid. It may be more convenient to choose a 128×128 or 129×129 domain to allow repeated subsampling.

Problem 9.5: Open-Ended Real-Data Problem — Multigrid

The strength of multigrid lies in its ability to solve poorly conditioned, nonlocal problems. Let's generate a set of estimates, such as those in Figure 5.17 on page 163, now using multigrid.

In particular, create a truth random field

$$Z_{x,y} = \frac{x^2 + y^2}{100} \qquad -50 \leq x, y \leq 50$$

from which we create noisy measurements

$$M = Z + V \qquad V \sim \sigma^2 I$$

with a noise variance of $\sigma^2 = 1$.

The analytical forms which preceded Figure 5.17 were expressions for P, whereas multigrid requires system matrix A, which is expressed in terms of P^{-1}. To avoid inverting a large matrix, we can choose P^{-1} directly by selecting a sparse/Markov prior, such as membrane or thin-plate, with an appropriately chosen correlation length, for example based on Figure 5.15.

Create a "satellite track" measurement pattern, such as that shown in Figure 5.17. Plot the estimates and discuss the number of multigrid iterations required for convergence, as a function of the choice of prior and the correlation length.

10

Kalman Filtering and Domain Decomposition

In this chapter we examine the solution of large dynamic estimation problems, problems which may stem from one of two sources:

1. Problems which are inherently spatio-temporal, such as the processing of video (spatial two-dimensional images over time) or remote sensing (the earth's surface or an atmospheric/oceanic volume sampled over time);

2. Multidimensional static problems, which have been converted to dynamic form, as discussed below.

In particular, rather than solving an inverse problem as one huge linear system of equations, as was undertaken in Chapter 9, here we want to benefit from the dynamic aspects of the problem to break it into smaller pieces.

Recall from Section 4.1.2 that we were able to show the algebraic equivalence or duality between a given dynamic problem

$$\underline{z}(t+1) = A(t)\underline{z}(t) + B(t)\underline{w}(t) \qquad \underline{w}(t) \sim \mathcal{N}(\underline{0}, I) \tag{10.1}$$

$$\underline{m}(t) = C(t)\underline{z}(t) + \underline{v}(t) \qquad \underline{v}(t) \sim \mathcal{N}\big(\underline{0}, R(t)\big) \tag{10.2}$$

and an associated static problem (from (4.24))

$$\underline{\bar{m}} = \bar{C}\underline{\bar{z}} + \underline{\bar{v}} \qquad \underline{\bar{z}} \sim (\underline{\bar{\mu}}, \bar{P}) \qquad \underline{\bar{v}} \sim (\underline{0}, \bar{R}) \; . \tag{10.3}$$

In that case the pieces of the dynamic problem, spread over time, were concatenated to construct a single static problem. Given that it was possible to construct such an equivalence, the obvious question is whether the process can be turned around: can we convert a static problem into a dynamic one?

There are two reasons why such a conversion to a dynamic problem offers an attractive alternative to the linear systems methods of Chapter 9:

P. Fieguth, *Statistical Image Processing and Multidimensional Modeling*, Information Science and Statistics, DOI 10.1007/978-1-4419-7294-1_10, © Springer Science+Business Media, LLC 2011

1. The Kalman filter produces both estimates and estimation error statistics. In some cases these error statistics are badly needed, for example to interpret the statistical significance of the estimation results, or possibly to further incorporate measurements via data fusion (as in Section 3.4). Although it is possible to acquire estimation error statistics from linear systems methods, in most cases computing these statistics is computationally prohibitive (except for the fully stationary/periodic case of the FFT).

2. If it is possible to break a large problem into relatively small pieces, there may be great computational advantages over solving the linear system as a whole. In particular, given an $N \times N$ random field, if it is possible to express the N columns of the random field as a dynamic process, each of length N, then (ignoring sparsity) we have the complexities

$$\text{Static Complexity } \mathcal{O}(N^6) \qquad \text{Dynamic Complexity } \mathcal{O}(N \cdot N^3). \tag{10.4}$$

   Thus the dynamic approach possibly offers a reduction in complexity of $\mathcal{O}(N^2)$ relative to direct static solvers.

The crux of the matter, then, is the following key question:

Do the portions of a given static problem obey a dynamic relationship?

Suppose we have a zero-mean random field $\underline{z} \sim P$, which we break into two pieces with known statistics

$$\underline{z} = \begin{bmatrix} \underline{z}_1 \\ \underline{z}_2 \end{bmatrix} \sim \begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix} \tag{10.5}$$

We can assert a linear dynamic relationship between $\underline{z}_1$ and $\underline{z}_2$:

$$\underline{z}_2 = A\underline{z}_1 + B\underline{w} \qquad E\left[\underline{z}_1 \underline{w}^T\right] = \mathbf{0} \qquad \underline{w} \sim I\,. \tag{10.6}$$

However, the relationship in (10.6) implies the cross-statistics between $\underline{z}_1$ and $\underline{z}_2$, therefore we should be able to learn A, B in (10.6) from the given, static cross-statistics in (10.5):

$$P_{12} \equiv E\left[\underline{z}_1 \underline{z}_2^T\right] = E\left[\underline{z}_1 (A\underline{z}_1 + B\underline{w})^T\right] \tag{10.7}$$

$$= P_{11} A^T + \mathbf{0} \tag{10.8}$$

therefore

$$A = P_{12}^T P_{11}^{-1} = P_{21} P_{11}^{-1}. \tag{10.9}$$

Similarly, from the covariance of $\underline{z}_2$:

$$P_{22} \equiv E\left[\underline{z}_2 \underline{z}_2^T\right] \tag{10.10}$$
$$= E\left[(A\underline{z}_1 + B\underline{w})(A\underline{z}_1 + B\underline{w})^T\right] \tag{10.11}$$
$$= AP_{11}A^T + BB^T \tag{10.12}$$

therefore

$$BB^T = P_{22} - AP_{11}A^T. \tag{10.13}$$

We see, therefore, that the static–dynamic conversion is possible. Barring computational limitations, the Kalman filter can then immediately be applied to the problem. This chapter surveys some of the more common ways in which dynamic methods are applied to large-scale estimation problems, with Section 10.2 in particular looking at sparse / reduced-order forms of the Kalman filter.

10.1 Marching Methods

Suppose we wish to solve a d-dimensional spatial problem, for which a spatial prior is given, using a Kalman filter to proceed dynamically along a sequence of $(d-1)$-dimensional slices; that is, row-by-row or column-by-column for a 2D problem, and plane-by-plane in 3D.

We are given a two-dimensional random field

$$Z = \left[\underline{z}_1\, \underline{z}_2 \cdots \underline{z}_N\right]. \tag{10.14}$$

To model this sequence of columns dynamically, we seek a linear model of the form

$$\underline{z}_{i+1} = f\left(\underline{z}_i, \underline{z}_{i-1}, \ldots, \underline{z}_1\right) + \underline{w}_i. \tag{10.15}$$

Ideally $\underline{z}_{i+1}$ would, for example, be a function of $\underline{z}_i$ only. From Chapter 6 we recognize this as a statement of Markovianity. Indeed, if Z is first- or second-order Markov,[1] then

$$E_{\text{LLSE}}\left[\underline{z}_{i+1} | \underline{z}_i, \underline{z}_{i-1}, \ldots, \underline{z}_1\right] = E_{\text{LLSE}}\left[\underline{z}_{i+1} | \underline{z}_i\right]. \tag{10.16}$$

Similarly, if Z is third- through fifth-order Markov, then

$$E_{\text{LLSE}}\left[\underline{z}_{i+1} | \underline{z}_i, \underline{z}_{i-1}, \ldots, \underline{z}_1\right] = E_{\text{LLSE}}\left[\underline{z}_{i+1} | \underline{z}_i, \underline{z}_{i-1}\right]. \tag{10.17}$$

Since we are seeking linear dynamic models, leading to linear estimators, we assert a Gauss–Markov autoregressive form,

$$\underline{z}_{i+1} = A_i^{(1)} \underline{z}_i + B_i w_i, \tag{10.18}$$

[1] See Figure 6.5 for the definition of Markov random field order.

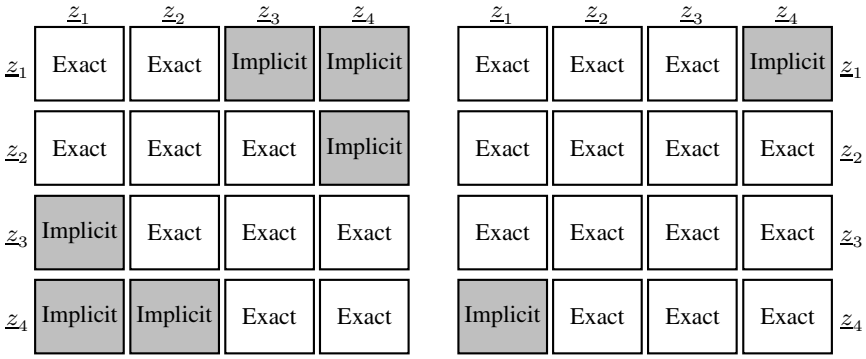


Fig. 10.1. The above panels show how the original static prior covariance is represented by the marching method. For each increment in marching order, one additional band on each side of the diagonal is modelled exactly. The success of the marching method depends on how well more distant bands can be approximated by the modelled statistics of the near-diagonal bands.

a first-order marching model corresponding to (10.16), or

$$\underline{z}_{i+1} = A_i^{(1)} \underline{z}_i + A_i^{(2)} \underline{z}_{i-1} + B_i w_i, \tag{10.19}$$

a second-order[2] marching model corresponding to (10.17). Clearly these generalize to the jth order case

$$\underline{z}_{i+1} = A_i^{(1)} \underline{z}_i + A_i^{(2)} \underline{z}_{i-1} + \cdots + A_i^{(j)} \underline{z}_{i-j+1} + B_i w_i. \tag{10.20}$$

We have already seen (Section 4.1.1) that such higher-order models can be cast into first-order Gauss–Markov form by state augmentation. That is, under the following change of variables

$$\bar{\underline{z}}_i = \begin{bmatrix} \underline{z}_i \\ \underline{z}_{i-1} \\ \vdots \\ \underline{z}_{i-j+1} \end{bmatrix} \qquad \bar{A}_i = \begin{bmatrix} A_i^{(1)} & A_i^{(2)} & \cdots & A_i^{(j)} \\ I & & & \\ & I & & \\ & & \ddots & \end{bmatrix} \qquad \bar{B}(t) = \begin{bmatrix} B_i \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{10.21}$$

then we obtain the standard first-order dynamic recursion

$$\bar{\underline{z}}(t+1) = \bar{A}(t) \bar{\underline{z}}(t) + \bar{B}(t) \underline{w}(t). \tag{10.22}$$

[2] One-dimensional and multidimensional Markov orders have differing definitions, so it is indeed correct that a second-order 1D marching process $\{\underline{z}_i\}$ corresponds to a third- or fourth-order 2D random field Z. This fluidity or convertibility of order we have seen before, in the autoregressive state augmentation of Section 4.1.1.

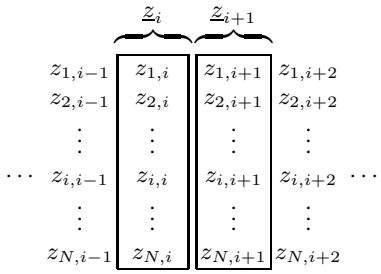


Fig. 10.2. Is the marching model modelling too much? Consider the first-order model of (10.18), as sketched here, which represents column $\underline{z}_{i+1}$ in terms of the previous column $\underline{z}_i$. This means, however, that we represent *all* of the joint statistics of the columns, including those of pixels $z_{1,i}$ and $z_{N,i+1}$, spatially separated by N pixels, whereas the relationship between $z_{1,i-1}$ and $z_{1,i+1}$, spatially separated by only two pixels, is *not* explicitly modelled.

That is, the Kalman filter can, with no modifications, solve first- and higher-order multidimensional marching problems.

For the $N \times N$ two-dimensional case, the computational complexity of computing estimates has been reduced from $\mathcal{O}(N^6)$ for direct solvers, to $\mathcal{O}\big(N \cdot (Nj)^3\big)$ for the jth order marching approach, from which we can clearly see the motivation for making j as small as possible. In the $N \times \cdots \times N$ case in d dimensions, the direct approach has complexity $\mathcal{O}\big(N^{3d}\big)$, whereas the marching approach has complexity $\mathcal{O}\big(N \cdot N^{3(d-1)} j^3\big)$.

There remain two questions:

1. What, exactly, do we lose in going to the dynamic/marching approach? The reduction in computational complexity must bring with it some limitations or approximations in the problem solution.

2. Is the marching method computationally feasible, and is it possible to reduce the state size further for additional computational benefits?

In terms of the former question, indeed, some approximations are made. The jth order dynamic model precisely represents the joint statistics of $j+1$ consecutive columns, however the joint statistics of columns more than j steps apart are modelled implicitly. Figure 10.1 illustrates this behaviour: each increment in j extends the exact representation of the prior covariance by one additional band. Whether the implicit statistics are correct depends on the Markov order of the random field.

In terms of the latter questions, consider Figure 10.2. The first-order marching model represents *all* of the joint statistics of the two columns, including those of distantly-

separated state elements $z_{1,i}$ and $z_{N,i+1}$, whereas the statistics of pixels only two columns apart, such as between $z_{1,i-1}$ and $z_{1,i+1}$, are only implicitly modelled. It would seem that representing the joint statistics of each column is excessive, and that only a portion of the column should somehow be needed: this is the goal of the strip-based, reduced-update, and reduced-order Kalman filters discussed in Section 10.2.

10.2 Efficient, Large-State Kalman Filters

At this point we assume that we have been given a large, dynamic estimation problem, whether inherently dynamic (a spatio-temporal problem) or not (a spatial problem, converted to a dynamic one via marching).

In any event, we have been given a dynamic process model and associated dynamic measurements and measurement model:

$$\underline{z}(t+1) = A(t)\underline{z}(t) + B(t)\underline{w}(t) \qquad \underline{w}(t) \sim \mathcal{N}(\underline{0}, I) \tag{10.23}$$

$$\underline{m}(t) = C(t)\underline{z}(t) + \underline{v}(t) \qquad \underline{v}(t) \sim \mathcal{N}\big(\underline{0}, R(t)\big), \tag{10.24}$$

where, by definition, the model of (10.23) is first-order Gauss–Markov (although possibly representing a higher-order spatial Markov process by state-augmentation (10.21)).

There are two basic reasons why it is much harder to find efficient dynamic algorithms compared to static ones:

1. In very many cases (e.g., membrane, thin-plate, and Markov priors) the static problem is characterized by very sparse matrices L, Q, however in the Kalman filter sparsity tends to disappear:

   - In the prediction step

   $$P(t+1|t) = A(t)P(t|t)A^T(t) + B(t)B^T(t) \tag{10.25}$$

   unless A is diagonal, generally $P(t+1|t)$ is *less* sparse than $P(t|t)$.

   - In the update step,

   $$K(t) = P(t|t-1)C^T \left\{CP(t|t-1)C^T + R\right\}^{-1}, \tag{10.26}$$

   the gain K will generally be a dense matrix, since the inverse of a sparse $(CPC^T + R)$ is normally dense.

2. In many cases the prior model for the static case is simple, stationary, sparse, or at least regularly structured; we have taken advantage of such attributes to develop

a variety of efficient approaches. For example, even the complex, nonstationary prior of Example 5.1 on page 159 admits an implicit, algorithmic implementation of the prior.

In the Kalman filter, on the other hand, even if the dynamic process starts with a simple, stationary, well-structured covariance P_o, over time the covariance $P(t|t)$ becomes complex, nonstationary, and unstructured, as was seen in Application 4, because the behaviour of $P(t|t)$ depends on the interaction of the problem dynamics with the location C and quality R of the measurements.

If the number of elements in $\underline{z}$ is modest, then a direct implementation of the KF is possible, treating all of the system matrices A, B and estimation error covariances $P(t|t-1), P(t|t)$ as dense.

There are two limitations to the straightforward application of the dense Kalman filter to large-scale problems:

1. Computational complexity, specifically due to matrix–matrix multiplication and matrix inversion;
2. Storage complexity, due to the storage of dense covariances.

A great many approaches have been developed for implementing Kalman filter–like solutions to large-scale dynamic estimation problems, in many cases customized to the specific attributes of a multidimensional estimation problem. Some of the most common approaches are summarized in the following sections:

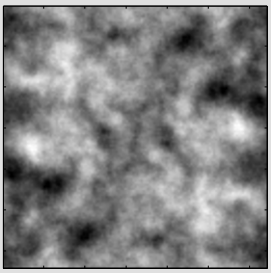
- Section 10.2.2 — Steady-State KF: do not update or predict covariances
- Section 10.2.3 — Strip KF: break the problem into pieces
- Section 10.2.4 — Reduced-Update KF: limit the extent of the model
- Section 10.2.5 — Sparse KF: implicit representation of covariances
- Section 10.2.6 — Reduced-Order KF: use basis reduction to reduce state size

10.2.1 Large-State Kalman Smoother

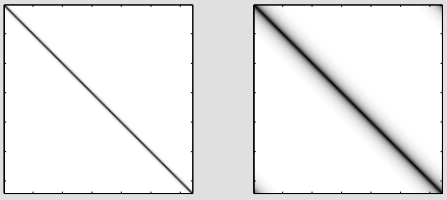
In many cases $\underline{z}(t)$ is not a causal process, particularly when Z is marching spatially in solving a multidimensional static problem. In such cases we may wish the estimates $\underline{\hat{z}}(t)$ to be computed acausally. In principle we have already discussed acausal filtering of dynamic processes using the RTS smoother in Section 4.3.3, however to compute smoothed estimates over $t \in [0, \tau]$ the RTS smoother must store $P(t|t)$ for every time t in the interval. In those cases where $\underline{z}$ is a large problem, the storage

Example 10.1: Interpolation by Marching

Suppose we have a stationary, periodic random field

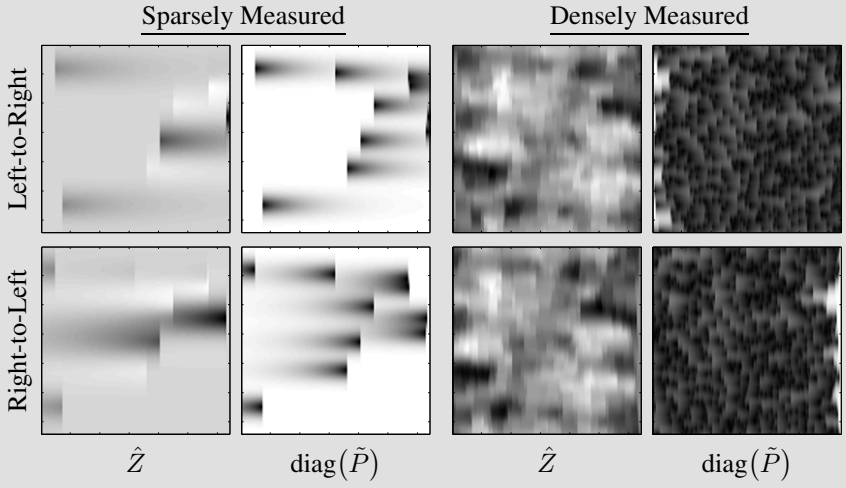


We have sparse measurements of the random field, therefore the FFT method cannot be used for estimation, so we use the Kalman filter and march, column by column. The column dynamics can be learned from the prior spatial statistics



Matrix A Matrix BB^T

where the sparse structure of A is consistent with the random field being Markov. With the model in place, we can use the Kalman filter to generate causal (left-to-right) or anticausal (right-to-left) estimates:



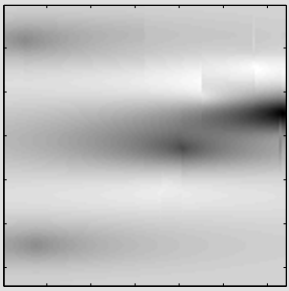
Example continues . . .

Example 10.1: Interpolation by Marching (cont'd)

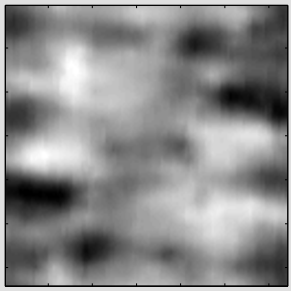
We can clearly see how the measurements and error variances respond unidirectionally to the measurements, especially in the sparsely-measured cases (left).

It is possible to consider merging the two separate (causal and anticausal) Kalman filter results, based on (10.28):

Sparsely Measured



Densely Measured



There are a number of artifacts present, which stem from merging the results on the basis of error *variances* only, rather than the full covariance (which would be too large to store in multidimensional problems of interest). Nevertheless the merged estimates are considerably more appealing, and probably more useful, than the unidirectional ones on the previous page.

of τ covariances $P(1|1), \ldots, P(\tau|\tau)$ is almost certainly prohibitive, leaving us with three alternatives:

1. Don't smooth, solve the estimates causally only. For many true temporal estimation problems this is likely to be appropriate, but less so for spatial-marching ones.

2. Perform suboptimal smoothing, whereby the covariances $P(t|t)$ are stored in banded / kernel / sparse inverse or some other compact form. Running an information Kalman filter (Section 4.3.1) in a sparse-matrix form, as discussed in Section 10.2.5, may indeed be practical.

3. Perform suboptimal smoothing by running the Kalman filter twice, once causally and once anticausally, and merging the two sets of results, as illustrated in Example 10.1. Thus we compute

$$\underline{\hat{z}}(t|0,\ldots,t), P(t|0,\ldots,t) \qquad \text{and} \qquad \underline{\hat{z}}(t|t,\ldots,\tau), P(t|t,\ldots,\tau) \tag{10.27}$$

which are merged as

$$\hat{z}_i(t|0,\ldots,\tau) = \frac{\hat{z}_i(t|0,\ldots,t)/P_{i,i}(t|0,\ldots,t) + \hat{z}_i(t|t,\ldots,\tau)/P_{i,i}(t|t,\ldots,\tau)}{-1/P_{i,i}(t) + 1/P_{i,i}(t|0,\ldots,t) + 1/P_{i,i}(t|t,\ldots,\tau)} \tag{10.28}$$

for example, where the $-1/P_{i,i}(t)$ term in the denominator is to subtract out the prior, which would otherwise be double-counted in the statistics. Computing the merged results still requires saving $\underline{\hat{z}}$ and $\text{diag}(P)$ at each point in time, for both directions, requiring a storage of 4τ times greater than $\underline{z}$ alone, although only four times greater than the storage of Z, the underlying static random field.

Practical solutions to the Kalman smoother therefore rely on efficient methods for Kalman filtering, discussed in the following sections.

10.2.2 Steady-State KF

If the dynamic system is temporally stationary, then the steady-state Kalman filter (Section 4.3.2) may be applied. Although large, matrices $P(t|t)$, $P(t|t-1)$, $K(t)$ are not a function of t and need to be calculated only once, leaving the relatively simple vector equations to recursively calculate $\underline{\hat{z}}(t|t), \underline{\hat{z}}(t|t-1)$. Since the vector equations explicitly depend only on K, and not on $P(t|t), P(t|t-1)$, it is actually only K which needs to be stored. Finally, because K does not need to satisfy positive-definiteness it may readily be approximated, for example using a sparse/banded approach (Section 5.3) to address storage complexity.

It should be pointed out that the *fully* stationary case, stationary in *all* dimensions with periodic boundaries, should be solved using FFT methods (Section 8.3). However, the steady-state Kalman filter requires only temporal stationarity, and *not* spatial stationarity or periodicity.

10.2.3 Strip KF

As was made clear in Figure 10.2, the regular Kalman filter will consider *all* of the interrelationships of elements in $\underline{z}$. If $\underline{z}$ contains a set of pixels, distributed spatially, such as the column or row of an image, then almost certainly the interrelationships of distantly-separated state elements are not of great value. Therefore it is logical to consider breaking the state into a number of pieces, which is the essence of the Strip Kalman Filter [337].

The state vector $\underline{z}(t)^T = \begin{bmatrix} z_1(t) & z_2(t) & \cdots & z_n(t) \end{bmatrix}$ is divided into a number of pieces

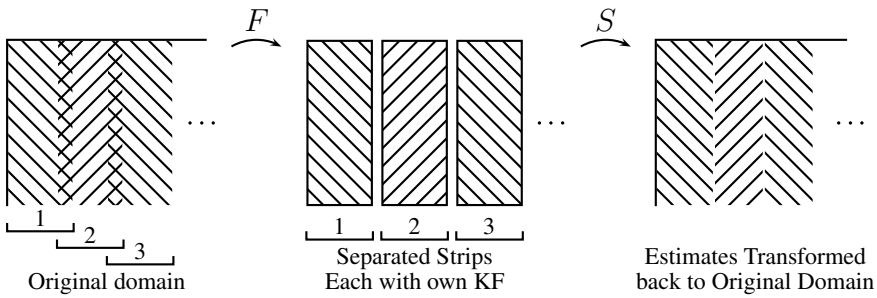


Fig. 10.3. The Strip Kalman Filter: The Kalman filter state vector, here the row of an image, is partitioned into strips, such that a separate Kalman filter is run on each strip, with the estimated strips then recombined.

$$\begin{aligned}
\underline{z}_1(t)^T &= \begin{bmatrix} z_1(t) & \cdots & z_w(t) \end{bmatrix} \\
\underline{z}_2(t)^T &= \begin{bmatrix} z_{w-\Delta}(t) & \cdots & z_{2w-\Delta}(t) \end{bmatrix} \\
&\vdots \\
\underline{z}_q(t)^T &= \begin{bmatrix} z_{(q-1)(w-\Delta)}(t) & \cdots & z_n(t) \end{bmatrix}
\end{aligned} \tag{10.29}$$

for some appropriate strip width w and strip overlap Δ, where

$$w \simeq \frac{n}{q} + \Delta. \tag{10.30}$$

The resulting strips are processed separately, and the resulting estimates from the individual strips concatenated to form an estimate of the original problem, as illustrated in Figure 10.3.

This process of dividing into pieces, independent processing, and recombining is exactly the same as what was described under local processing in Section 8.2.3. In particular, compare Figure 10.3 with the overlapped transformation of (8.49) and (8.51) on page 259.

Because the computational complexity of the Kalman Filter goes as the cube of the state size, whereas the original filter had a complexity of $\mathcal{O}(n^3)$ per row, the strip filter has a complexity of

$$q \cdot \left(\frac{n}{q} + \Delta \right)^3 \tag{10.31}$$

per row, where q represents the number of strips, and Δ the overlap between the strips. Differentiating (10.31), we can find the optimum number of strips, minimizing computational complexity, as

$$q_{opt} = \frac{2n}{\Delta} \tag{10.32}$$

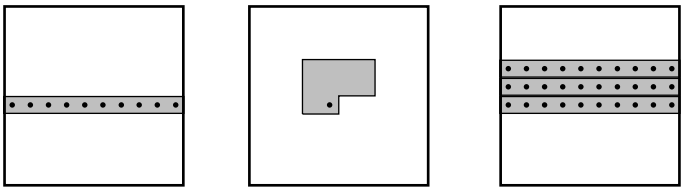


Fig. 10.4. The Reduced Update principle: Suppose we wish to produce estimates for a given row (a), based on some local half-plane model (b). However, under a marching scheme, even to estimate just that single scalar would require augmenting the row-state with multiple previous rows (c) to preserve the necessary history, a significant computational complexity. The RUKF performs the state update only within the model support in (b).

although there may be other factors — density of measurements, appearance of estimation artifacts, spatial correlation length — which constrain the appropriate range of w and Δ.

10.2.4 Reduced-Update KF

The one disadvantage of the strip-based Kalman filter is that by processing the strips independently, if the measurements are sufficiently sparse the boundaries of the strips will necessarily appear as artifacts in the estimates. Instead, the Reduced Update Kalman filter [188, 337, 338] produces state estimates by scanning along the state vector, one scalar element at a time.

The principle of the RUKF is illustrated in Figure 10.4. Suppose that each image row contains N state elements, and that the 2D auto-regressive model in Figure 10.4(b) has a size of $q \times q$ pixels. In terms of implementing a scalar, raster-scanning Kalman filter:

- The *prediction* step is based only on the size of the model in Figure 10.4(b), involving $\mathcal{O}(q^2)$ state elements;
- The *update* step, on the other hand, needs to update all of the elements in the augmented state in Figure 10.4(c), involving $\mathcal{O}(qN)$ state elements.

Since we expect that N is somewhat larger than q, and because computational complexity in the Kalman filter goes as the cube of the number of state elements, it is clear from the above two points that the prediction step is relatively straightforward, and that efforts towards simplification should be concentrated on the update step.

The goal of the RUKF is to perform the update step, but involving only those $\mathcal{O}(q^2)$ state elements in the model. Because these q^2 elements are those most likely to be meaningfully related to the measurement being processed, the degree of approximation is modest.

Adapting the notation from [188], we can summarize the RUKF in standard Kalman filter form as follows. Suppose we are given a stationary, causal Markov model as in (6.38):

$$z_t = \sum_{s \in \mathcal{N}} g_s z_{t-s} + w_t, \tag{10.33}$$

where $\mathcal{N}$ is a symmetric half-plane neighbourhood, t is a state index (here two-dimensional), and w is a driving process noise with stationary variance σ^2. Then the RUKF becomes

Prediction:

$$\hat{z}(t|t-1) = \sum_{s \in \mathcal{N}} g_s \hat{z}(t-s|t-1) \tag{10.34}$$

$$\bar{P}(t,i|t-1) = \sum_{s} g_s P(t-s,i|t-1) \quad i < t \tag{10.35}$$

$$\bar{P}(t,t|t-1) = \sum_{s} g_s P(t,t-s|t-1) \tag{10.36}$$

Update:

$$\hat{z}(i|t) = \hat{z}(i|t-1) + k(t-i|t)\big(m(t) - \hat{z}(t|t)\big) \tag{10.37}$$

$$P(i,j|t) = \bar{P}(i,j|t-1) - k(t-i|t)\bar{P}(t,j|t-1) \tag{10.38}$$

$$k(i|t) = \bar{P}(t,i|t-1) \cdot \big(\bar{P}(t,t|t-1) + r(t)\big)^{-1}. \tag{10.39}$$

The structural similarity of the above RUKF to that of the standard Kalman filter in Chapter 4 is clear.

Although a causal model is used, it is important to understand that the produced estimates are not causal: a given measurement $m(t)$ is updated anti-causally into the neighbourhood $\mathcal{N}$ preceding t, and will be predicted causally into the state elements following t. Of course the degree of acausality is limited by the size of $\mathcal{N}$.

10.2.5 Sparse KF

Rather than breaking the problem into pieces (strip KF) or localizing the state (RUKF), one alternative is to take advantage of problem sparsity (Section 5.3):

- Reduced storage complexity
- Reduced matrix–matrix multiplication complexity
- Possibly, reduced matrix inversion complexity

Of the matrices which appear in the Kalman filter, it is common for C, B, R to be sparse. Our focus is therefore on the dynamics matrix A and the error covariances $P(t|t), P(t|t-1)$:

Dynamics: If the time-discretization is chosen sufficiently fine, then for most physical systems $A(t)$ will be sparse and nearly diagonal since a given state element z_i is, in a brief instant of time, likely to influence only those few other elements in its immediate vicinity.

Covariances: Unless the dynamics A are diagonal, over time all of the elements of $\underline{z}$ become correlated and $P(t|t), P(t|t-1)$ become full. Sparsity is therefore a matter of assertion or approximation.

The estimation errors do tend to be *locally* correlated, however the correlation decays slowly with offset, and the covariances $P(t|t), P(t|t-1)$ are not naturally sparse. In many cases it may be possible to assume a Markov model for the errors, however, meaning that it is $P^{-1}(t|t)$ for which a sparse form may be a reasonable approximation.

A variety of sparse Kalman filters has been proposed [10, 65, 66, 307], most of them [65, 66, 307] based on sparsifying the inverse covariance, therefore actually implementing a sparse version of the Information Kalman filter (Section 4.3.1). We need to specify two things:

1. A sparsification operation, normally banded-Markov,
2. A sparse-matrix inversion operation.

Under the assumption that the covariances satisfy diagonal dominance, the series approximation to matrix inversion (5.21) can be used to efficiently implement the latter matrix inversion.

The former sparsification operator seems straightforward in principle: for information (inverse) covariances, set to zero all elements outside of some set of bands. In practice, the need to preserve positive-definiteness makes the operation more subtle.

10.2.6 Reduced-Order KF

Because much of remote sensing is inherently global and time-dynamic in nature, such as the discussion in Application 4 on page 122, there has been a particularly large number of Kalman filters implemented for large-scale time-dynamic data assimilation, such as in [103, 121, 191] and in citations [2–12] within [111].

The Reduced-Order Kalman filter [7, 103, 239, 287] seeks a reduction of basis, as discussed in Section 8.2.2 and as illustrated in Application 8 on page 285, in order

to reduce the dimensionality of the state to an extent that a direct implementation of the Kalman filter for the reduced state is feasible.

From (8.117), suppose that some operator $\mathcal{A}$ characterizes the discrete-time dynamics of a random process $\underline{z}'$:

$$\underline{z}'(t+1|t) = \mathcal{A}\big(\underline{z}'(t|t)\big). \tag{10.40}$$

In many cases, the dynamic problem involves fluctuations about a mean (Example 3.3); for example, the ocean sea-surface temperature exhibits modest variations about a relatively complex mean. If we subtract out the possibly time-varying mean $\underline{z}'_o(t)$,

$$\underline{z}(t) = \underline{z}'(t) - \underline{z}'_o(t) \tag{10.41}$$

then $\underline{z}$ is our process of interest. We can define a reduced state

$$\bar{\underline{z}}(t) = F\underline{z}(t), \tag{10.42}$$

where F is not obligated to preserve any (possibly sharp and complex) details of $\underline{z}'_o$, only the typically smoother deviations $\underline{z}$ about the mean. The degree of reduction, the number of rows in F, is chosen such that a regular, dense Kalman filter can be applied to $\bar{\underline{z}}$.

Given updated estimates, the estimates of $\underline{z}'$ are found by inverting (10.41),(10.42):

$$\hat{\underline{z}}'(t|t) = \underline{z}'_o + S\hat{\bar{\underline{z}}}(t|t), \tag{10.43}$$

where S is the pseudoinverse of F, as in Section 8.2.2.

The high resolution estimates are passed through dynamics $\mathcal{A}$, meaning that the prediction step takes place in the *unreduced* domain:

$$\hat{\bar{\underline{z}}}(t+1|t) = F\Big[\mathcal{A}\big(\underline{z}'_o(t) + S\hat{\bar{\underline{z}}}(t|t)\big) - \underline{z}'_o(t+1)\Big]. \tag{10.44}$$

In the event that $\mathcal{A}$ is linear, then the deterministic and stochastic portions of the problem decouple (the Wold decomposition [248]), implying that the Kalman filter can operate mean-removed, as usual, and allowing the predict step to take place in the reduced domain:

$$\hat{\bar{\underline{z}}}(t+1|t) = \underbrace{FAS}_{\bar{A}}\hat{\bar{\underline{z}}}(t|t). \tag{10.45}$$

10.3 Multiscale

All of the other approaches in this chapter have synthesized a dynamic problem by cutting a static problem into spatial pieces, such that the "time" variable t of the Kalman filter essentially indexes the rows, columns, or planes of a spatial problem.

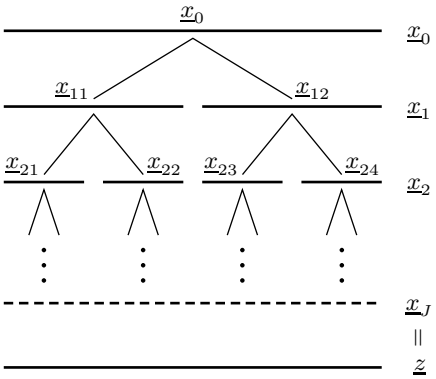


Fig. 10.5. A dyadic representation of a one-dimensional process $\underline{z}$: the coarse representation $\underline{x}_0$ is split, such that $\underline{x}_1$ is broken into two decoupled parts $\underline{x}_{11}, \underline{x}_{12}$. The repeated splitting continues for J scales; the goal is to find a tree model such that the finest scale $\underline{x}_J$ possesses the statistics of the represented process; that is, such that $\mathrm{cov}(\underline{x}_J) = \mathrm{cov}(\underline{z})$.

There are, to be sure, *other* ways of cutting up a spatial problem (see Problem 10.1, for example) than just rows or columns. As a significant departure from the rest of this chapter, we here consider allowing the Kalman filter "time" to index *scale*, rather than *space*. The basic idea, then, is to design a Kalman filter and RTS smoother to solve a static problem by iterating back and forth in scale. Intuitively this is vaguely analogous to the multigrid method of Section 9.2.5, except that multigrid is an iterative method, taking repeated passes at an estimation problem, whereas the Kalman filter / RTS smoother involve a *single* forwards and single backwards pass.

We know from Section 8.5 that subsampling or a change of resolution destroys the Markovianity of a random field, so the reader may wonder what a multiscale Kalman filter has to offer. It is pertinent, at this point, to clarify a distinction between multiresolution and multiscale:

A MULTIRESOLUTION model involves representing a random field with state elements that have varying spatial extent, such that coarse-scale pixels have a larger region of support than fine-scale ones.

A MULTISCALE model represents a random field on a hierarchy, but the nature of the representation at coarse scales does not necessarily involve low-resolution elements.

Therefore certain phenomena in remote sensing, such as $1/f$ power-laws [281] which involve structures at multiple scales, may be better served with a multiresolution approach, whereas textures and single-scale random fields which are spatially

Markov may be more appropriately represented via a multiscale model. The multiscale statistical model [69, 105, 169, 214] described in this section includes both Markov/multiscale and multiresolution models, although we focus on the GMRF case.

Suppose we are given the usual static problem

$$\underline{z}_{\text{static}} \sim P_{\text{static}} \qquad \underline{m} = C\underline{z}_{\text{static}} + \underline{v} \quad \underline{v} \sim R. \tag{10.46}$$

To construct an efficient multiscale model for $\underline{z}$, we seek to define a dynamic model

$$\underline{x}_{t+1} = A_t\underline{x}_t + B_t\underline{w}_t \qquad \underline{x}_0 \sim P_0 \quad \underline{w}_t \sim I \tag{10.47}$$

such that the dynamic multiscale state $\underline{x}$ *evolves* to $\underline{z}$ after a finite number of steps J. That is, we need to find A_t, B_t such that the finest scale of the multiscale problem equals or approximates the given static problem, such that

$$\begin{aligned} P_{\text{static}} \simeq P_J &= A_J P_{J-1} A_J^T + B_J B_J^T & (10.48)\\ &= A_J \left[A_{J-1} P_{J-2} A_{J-1}^T + B_{J-1} B_{J-1}^T\right] A_J^T + B_J B_J^T & (10.49)\\ &= \cdots \end{aligned}$$

and that, ideally, A_t and B_t have some particular form which makes estimation particularly easy and efficient.

Since the original static problem is equivalent to the finest scale, this is the only time-step at which the dynamic problem has measurements:

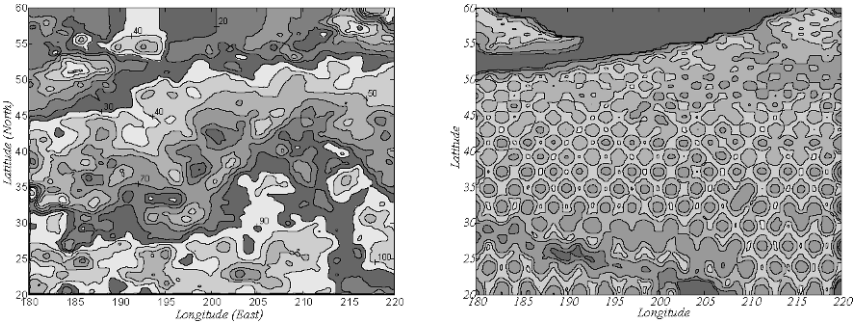
$$\underline{m}_{\text{static}} \equiv \underline{m}_J = C\underline{x}_J + \underline{v} \qquad \underline{v} \sim R. \tag{10.50}$$

Figure 10.5 illustrates the key idea to making this model efficient. The state is repeatedly divided into decorrelated pieces, such that $\underline{x}_{22}$ can depend on $\underline{x}_{11}$, but not on $\underline{x}_{12}$. This decoupling or decorrelation places a restriction on the dynamic model, such that A, B take the form

$$\underline{x}_1 = \begin{bmatrix}\underline{x}_{11}\\ \underline{x}_{12}\end{bmatrix} = A_1\underline{x}_0 + B_1\underline{w}_1 = \begin{bmatrix}A_{11}\\ A_{12}\end{bmatrix}\underline{x}_0 + \begin{bmatrix}B_{11} & \\ & B_{12}\end{bmatrix}\begin{bmatrix}\underline{w}_{11}\\ \underline{w}_{12}\end{bmatrix} \tag{10.51}$$

$$\underline{x}_2 = \begin{bmatrix}\underline{x}_{21}\\ \underline{x}_{22}\\ \underline{x}_{23}\\ \underline{x}_{24}\end{bmatrix} = A_2\underline{x}_1 + B_2\underline{w}_2 = \begin{bmatrix}A_{21} & \\ A_{22} & \\ & A_{23}\\ & A_{24}\end{bmatrix}\begin{bmatrix}\underline{x}_{11}\\ \underline{x}_{12}\end{bmatrix} + \begin{bmatrix}B_{21} & & & \\ & B_{22} & & \\ & & B_{23} & \\ & & & B_{24}\end{bmatrix}\begin{bmatrix}\underline{w}_{21}\\ \underline{w}_{22}\\ \underline{w}_{23}\\ \underline{w}_{24}\end{bmatrix} \tag{10.52}$$

Since the process noise $\underline{w}$ terms are white, the above large dynamic equations can be broken down into smaller pieces; for example,



Multiscale State Estimates Multiscale Error Variances

Fig. 10.6. Multiscale estimates and error variances [105]. The estimates are computed from the satellite altimetry data of Figure 1.3, where the higher accuracy of the estimates near the measured paths can be seen very clearly in the pattern of the error variances.

$$\begin{aligned} \underline{x}_{11} &= A_{11}\underline{x}_0 + B_{11}\underline{w}_{11} \\ \underline{x}_{21} &= A_{21}\underline{x}_{11} + B_{21}\underline{w}_{21} \\ \underline{x}_{24} &= A_{24}\underline{x}_{12} + B_{24}\underline{w}_{24}. \end{aligned} \tag{10.53}$$

Rather than the cluttered indices present in (10.53), we can generalize (10.53) in terms of a single index s,

$$\underline{x}(s) = A(s)\underline{x}\big(\uparrow s\big) + B(s)\underline{w}(s), \tag{10.54}$$

where $\uparrow s$ is the index of the parent of s. This generalized structure applies to one-dimensional dyadic trees, two-dimensional quad-trees, and indeed to *any* tree structure in any number of dimensions.

If we define each state to be some linear function of the underlying random field:

$$\underline{x}(s) = \Xi(s)\underline{z} \tag{10.55}$$

then the statistics at the finest scale allow us to infer the multiscale statistics,

$$\underline{z} \sim P_{\text{static}} \quad \Longrightarrow \quad P(s) = \text{cov}\big(\underline{x}(s)\big) = \Xi(s)P_{\text{static}}\Xi(s)^T \tag{10.56}$$

and from those the model parameters in (10.54):

$$A(s) = \big\{\Xi(s)P_{\text{static}}\Xi\big(\uparrow s\big)^T\big\} \cdot \big\{\Xi\big(\uparrow s\big)P_{\text{static}}\Xi\big(\uparrow s\big)^T\big\}^{-1} \tag{10.57}$$

$$B(s)B^T(s) = P(s) - A(s)P\big(\uparrow s\big)A^T(s). \tag{10.58}$$

It is not a coincidence that these equations appear very similar to the marching dynamics of (10.9), (10.13). The multiscale estimator is essentially a distributed marching algorithm, marching over the scales of a tree, rather than across space.

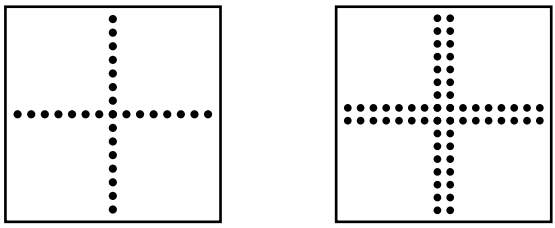


Fig. 10.7. If the underlying prior model is Markov, then keeping the pixellated values is sufficient to conditionally decorrelate the process into four quadrants. If each of those quadrants is further broken down into quadrants, we get a quad-tree whose finest scale statistics precisely equal the original Markov model. The number of rows or columns to keep in the state is a function of the Markov order of the underlying prior.

The multiscale estimation algorithm [69,105,169,214] is essentially like the Kalman smoother on the tree structure, with two exceptions:

1. In order for any measurement at any tree index to be able to influence the estimate at any other tree index, it is necessary to run the Kalman smoother in *reverse*, first filtering from fine-to-coarse, and then smoothing from coarse-to-fine.

2. In the tree a given node can have multiple child nodes, thus some sort of merge step is required to combine the information from multiple children in computing the estimate at the parent.

The resulting estimator performs an upwards pass, from fine to coarse, producing estimates $\underline{\hat{x}}_u(s)$ with uncertainty $\tilde{P}_u(s)$, followed by a downwards pass (the RTS smoother), producing estimates $\underline{\hat{x}}(s)$ with covariance $\tilde{P}(s)$. An example of multiscale estimates and estimation error variances is shown in Figure 10.6.

If $x(s) \in \mathbb{R}^{n(s)}$, then the complexities of the multiscale approach are

$$\mathcal{O}\left(\sum_s n^2(s)\right) \text{ Storage} \qquad \mathcal{O}\left(\sum_s n^3(s)\right) \text{ Computation.} \tag{10.59}$$

The remaining question is how to effectively select the state $\underline{x}(s) = \Xi(s)\underline{z}$ in order to satisfy the two multiscale objectives:

1. $n(s)$ is small at each tree index, for computational efficiency;

2. The finest-scale multiscale statistics P_J equal the given statistics P_{static}.

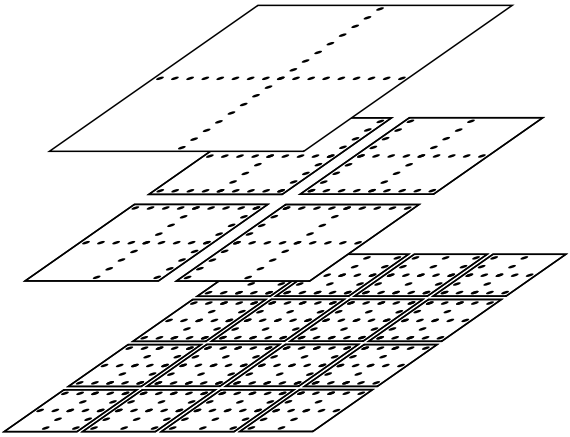


Fig. 10.8. Three levels of a multiscale hierarchy, with states chosen appropriate for a first-order Markov field, as in Figure 10.7. The purpose of each state is to hold that information which allows its four quadrants to be conditionally decorrelated.

Suppose we are given a two-dimensional static problem with a first-order Markov prior. Then if $\Xi(0)$ selects all of the pixels in the middle row and column, as illustrated in Figure 10.7, then conditioned on the root state $\underline{x}(0)$ the four quadrants are conditionally decorrelated, precisely the decoupling required in (10.51), and philosophically clearly related to the nested dissection of Section 9.1.3.

There is, however, no reason to content ourselves with limiting the decomposition of the field into four quadrants. We can proceed further, creating boundaries similar to those in Figure 10.7 within each of the quadrants, thus we can continue the successive subdivision of the field into smaller pieces, as illustrated in Figure 10.8.

The computational complexity is dominated by the large state at the tree root, with the complexity geometrically decreasing with scale. The computational complexity is shown as a function of dimension in Table 10.1.

A variety of generalizations may be considered:

- The point measurements of the original static problem are normally associated with the individual pixels at the finest level of the tree, however with the appropriate definition of $x(s)$ at coarser levels of the tree, nonlocal measurements can also be accommodated.

Dimensions	Problem Size	# Pixels	Multiscale Complexity
1D	n	$N = n$	$\mathcal{O}(n) = \mathcal{O}(N)$
2D	$n \times n$	$N = n^2$	$\mathcal{O}(n^3) = \mathcal{O}(N^{1.5})$
3D	$n \times n \times n$	$N = n^3$	$\mathcal{O}(n^6) = \mathcal{O}(N^2)$

Table 10.1. Computational complexity of the multiscale method, as compared to a direct solver, as a function of dimensionality.

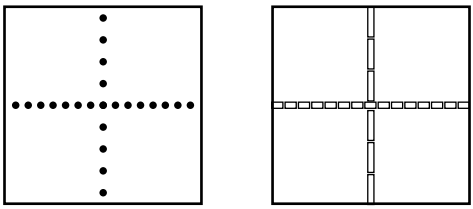


Fig. 10.9. The boundaries of Figure 10.7 may be much more densely sampled than needed. It may be a very reasonable approximation to subsample (left) or average (right) along the quadrant boundaries. The four quadrants will no longer be perfectly decorrelated, although very nearly so. Sample results, based on a subsampled state, are shown in Figures 10.10 and 10.11.

- It is possible to benefit from further computational efficiency via approximations. In particular, if the random field has a long correlation length, then it may be highly redundant (and poorly conditioned) to preserve *every* pixel along the quadrant boundaries. Instead, it may be very reasonable to select the state by subsampling or averaging [213, 232], as illustrated in Figure 10.9.

- The overlapped approach of Section 8.2.3 lends itself particularly well to the multiscale environment [169], especially when the states are approximated, rather than exact. With approximate states the quadrant decorrelation is imperfect, resulting in estimation inconsistencies and artifacts along the quadrant boundaries; by overlapping these artifacts can be reduced, as illustrated in Figure 10.11.

- Because the update step of the Kalman filter is essentially a static estimate, it is possible to use the multiscale estimator to solve the update step of large Kalman filters [191]. Indeed, the update step in the dynamic estimator in Application 4 was based on the multiscale method of this section.

It is possible to calculate both prior and posterior samples from the multiscale model, which is discussed in Section 11.2.3.

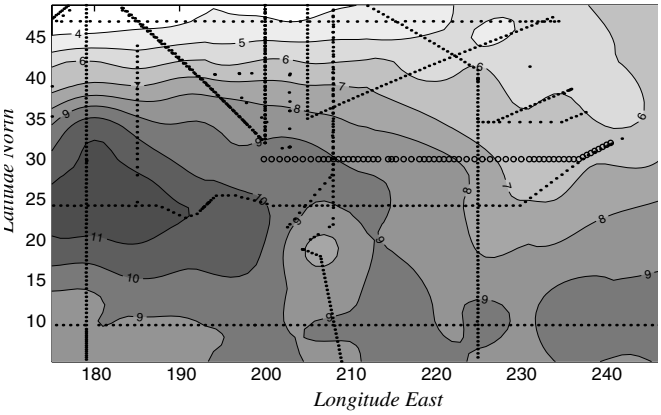
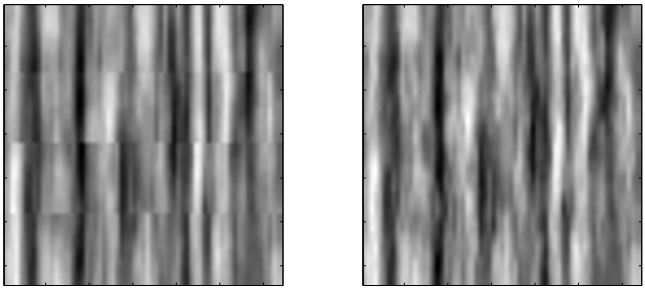


Fig. 10.10. A comprehensive, multidimensional estimation example [232]. Ship-based ocean temperature measurements are taken, as was illustrated in Figure 1.3 on page 5. The measurements are densely sampled in depth and the problem is essentially statistically stationary in latitude and longitude, therefore the problem was decoupled into multiple depth slices, as discussed in Example 8.1 on page 252. For each depth slice, a first-order tree can be created, as sketched in Figure 10.8. To reduce computational complexity and improve conditioning, a subsampled state is used, as in Figure 10.9.



Highly reduced-order model, with apparent texture artifacts

Even further reduced order, but overlapped

Fig. 10.11. Multiscale estimation of random-field textures with reduced-order models, with states as in Figure 10.9. The effectiveness of an overlapped approach, as discussed in Section 8.2.3, in reducing quadrant-boundary artifacts is clear.

Application 10: Video Denoising [178]

We wish to denoise video. Since successive images in a video are highly related a temporal filter seems reasonable, so we wish to implement something like a Kalman filter. However, the images are large and we want real-time filtering, so a standard Kalman filter in the spatial domain is not practical.

Video denoising approaches in the spatial domain [9, 41, 192] can be divided into three classes:

TEMPORAL-ONLY: An approach utilizing only the temporal correlations, neglecting spatial information.

SPATIAL-ONLY: Apply 2D spatial denoising to each video frame, taking advantage of the vast image denoising literature, but ignoring the temporal correlations.

SPATIO-TEMPORAL: More sophisticated methods exploiting both spatial and temporal correlations, such as simple adaptive weighted local averaging, 3D Kalman filtering, and 3D Markov models.

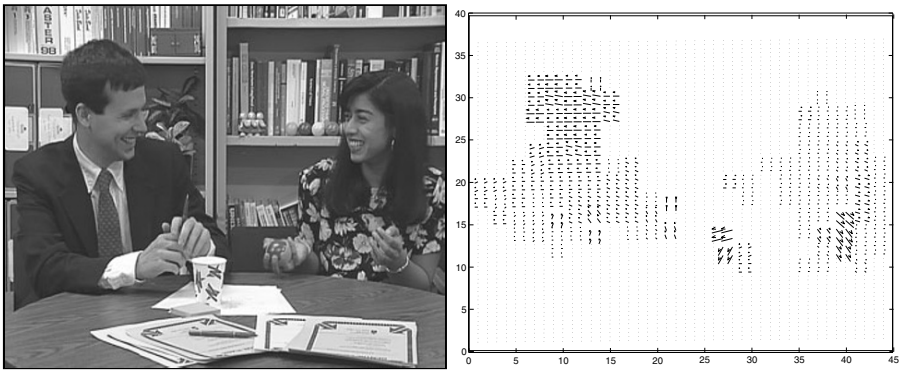
As an alternative to spatial-domain processing, we consider wavelet-based approaches, which have led to impressive results in 2D denoising. It would seem natural to select 3D wavelets for video denoising, however there are a number of drawbacks:

1. There is a clear asymmetry between space and time; for effective denoising we need to treat the space and time axes distinctly, not as a large 3D cube of data.
2. *All* of the image frames need to be in place in order to apply the 3D wavelet, therefore there is a long latency time between acquiring and denoising an image.
3. 3D wavelets cannot be sensitive to all of the possible object motions.

We can address these drawbacks by applying the 2D wavelet transform to each 2D image frame, and then performing spatio-temporal video filtering in the wavelet domain. That is, essentially we want to develop a large Kalman filter in the wavelet domain.

Next, we wish the temporal dynamics to correspond to motion. For most wavelets image motion does *not* imply a motion of coefficients in the wavelet domain, so we need to choose a shift-invariant, overcomplete wavelet transform [222]. The benefits of such an approach are clear:

1. The recursive, frame-by-frame approach implies low latency;
2. The wavelet decorrelative property allows very simple, fast, scalar temporal filtering;



Original Video Frame Motion for Prediction Step

Fig. 10.12. A frame from the *Paris* video, showing the inferred motion which is used as the time-prediction step in the wavelet domain.

3. Where motion estimates are unreliable, spatial (non-temporal) methods can provide denoising.

Given a noisy image sequence

$$\underline{m}(t) = \underline{z}(t) + \underline{v}(t), \tag{10.60}$$

we transform it into the wavelet domain

$$\bar{\underline{m}}(t) = W\underline{m}(t) = W\underline{z}(t) + W\underline{v}(t) = \bar{\underline{z}}(t) + \bar{\underline{v}}(t), \tag{10.61}$$

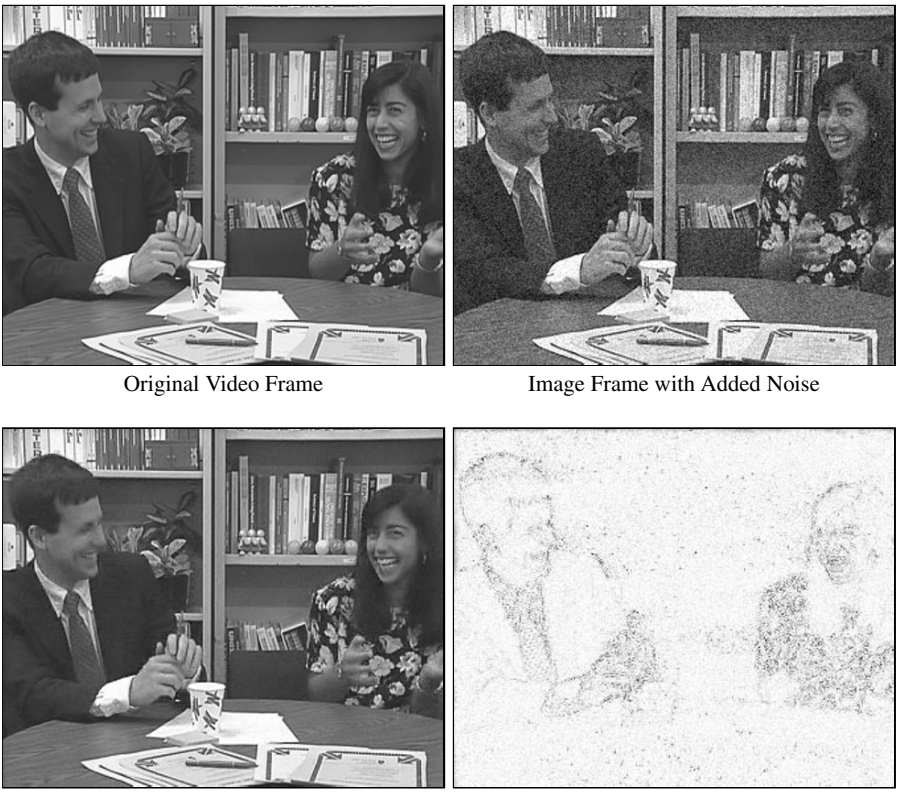
where the explicit transformation of the problem is possible because each image is densely sampled. We assert an autoregressive form for the signal model to fit the Kalman filter Gauss–Markov dynamics:

$$\bar{\underline{z}}(t+1) = A(t)\bar{\underline{z}}(t) + B(t)\bar{\underline{w}}(t) \tag{10.62}$$

for some white, stochastic driving process $\bar{\underline{w}}$. The inference of A and B is simplified here by assuming that each frame is related to its predecessor, subject to some displacement field $\underline{D}(t)$. Since the selected wavelet is shift-invariant, the wavelet coefficients are subject to the same motion as the image itself, such as those illustrated in Figure 10.12, thus the dynamic model simplifies as

$$\bar{Z}_{\underline{i}}(t) = \bar{Z}_{\underline{i}+\underline{D}_{\underline{i}}}(t-1) + 0 \cdot \bar{\underline{w}}(t). \tag{10.63}$$

This model captures only translations, and not occlusion or zooming. One can choose to *test* model (10.63) by hypothesis testing, and where the model is invalid (that is,



Original Video Frame | Image Frame with Added Noise

Wavelet–Kalman Denoised | Difference between Original and Denoised

Fig. 10.13. Applying the spatio-temporal wavelet denoising method of [178]: Wavelet artifacts are not apparent in the reconstruction and the error image is small, with great noise reduction in those parts of the images, away from edges, which can be reliably predicted over time.

where $Z(t-1)$ and $Z(t)$ cannot be matched by translation), then the null model can be asserted:

$$\bar{Z}_{\underline{i}}(t) = 0 \cdot \bar{Z}_{\underline{i}}(t-1) + B(t) \cdot \underline{\bar{w}}(t). \tag{10.64}$$

This model has no dynamics, and is thus a purely spatial problem, to which standard image denoising methods (Appendix C) can be applied.

The above approach was developed, implemented, and applied to video [178], with results as shown in Figure 10.13. For a relatively simple idea — using a Kalman filter in the wavelet domain with the prediction step based on motion estimation — the results are quite compelling.

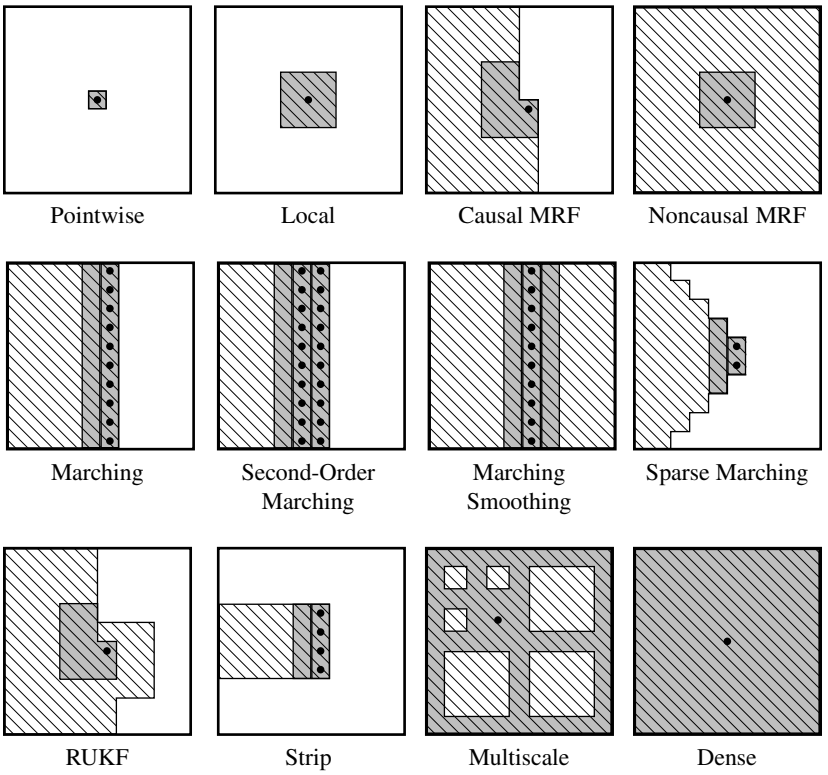


Fig. 10.14. An overview and visual comparison of estimation methods. The state (black dots) is written in terms of a model over some portion of the domain (shaded), however the estimated state may be influenced by measurements within some larger domain (hatched).

Summary

Figure 10.14 visually compares twelve methods of estimation. The models differ primarily on the basis of locality, causality, and order / complexity.

Clearly this list is not exhaustive, and should be understood to be complementary to other methods and alternatives already explored in this text:

- The Kalman filter algorithmic alternatives in Section 4.3,
- The methods of transformation and dimensionality reduction in Figure 5.2 at the start of Chapter 5,
- The methods of representation in Figure 6.13 at the end of Chapter 6.

For Further Study

The reader may find it interesting to follow the evolution of the Kalman filter, for spatial processing purposes, from the development of the original filter in 1960 [182], the Kalman smoother in 1965 [266], the two-dimensional strip filter in 1977 [337], the reduced update Kalman filter in the early 1980s [188, 338], the reduced order Kalman filter in 1989 [7, 287], and more recent work in three-dimensional and video filtering [192].

Sample Problems

Problem 10.1: Other Approaches to Marching

The marching method of Section 10.1 describes the solution of a two-dimensional problem by marching column-by-column. However, there is nothing special about breaking an image into columns.

Describe how you could perform 2D static estimation by dynamically marching diagonal by diagonal.

Problem 10.2: Marching Limitations

There are a number of limitations to the first-order, causal marching method. For each of the following limitations, briefly discuss or suggest an alternative:

(a) The computed estimates are causal, depending only on measurements in the current and previous columns.
(b) The first-order dynamic model is a poor approximation of the underlying prior statistics.
(c) The computational complexity is too high when each column has very many elements.
(d) The marching method assumes Markovianity; what do I do for a stationary non-Markov problem?

Problem 10.3: 2D Marching

Let $Z = [\underline{z}_1 \ \ldots \ \underline{z}_{64}]$ be a 64×64 two-dimensional zero-mean stationary process with periodic boundary conditions.

Let $\mathcal{G}$ be the "Tree-Bark" kernel from Example 6.1 on page 190. We consider two possible prior models for Z:

$$\mathcal{G}_1 = \mathcal{G} \quad \text{and} \quad \mathcal{G}_2 = (\mathcal{G})^T,$$

where $(\mathcal{G})^T$ is just the transpose of the kernel itself. Using the FFT method we can invert the model kernel $\mathcal{G}_1$ to find the covariances

$$P_{\text{self}} = \text{cov}(\underline{z}_i) \qquad P_{\text{cross}} = E\left[\underline{z}_i\, \underline{z}_{i+1}^T\right]$$

Note that because of the stationarity of Z, P_{self} and P_{cross} are not a function of i.

We will be marching column by column. From P_{self} and P_{cross} determine A, BB^T, the matrices in the first-order dynamic model. Initialize the Kalman filter with

$$\hat{\underline{z}}(1|0) = \underline{0} \qquad P(1|0) = P_{\text{self}}.$$

Use this Kalman filter to answer the following:

(a) Suppose we observe only a single pixel

$$10.0 = m = Z(20, 20) + v \qquad v \sim \sigma^2 = P_{\text{self}}(1, 1).$$

Use the Kalman filter to compute estimates and error variances; plot and interpret the results.

(b) Could the FFT method have been used to compute the estimates and error variances in part (a)?

(c) Now suppose we observe *every* pixel. Use the FFT method, based on prior $\mathcal{G}_1$, to create a random sample image M, which we will use as our observations:

$$M = Z + V \qquad [V]_{:} \sim R, \quad R_{i,j} = \delta_{i,j} P_{\text{self}}(1, 1).$$

Use the Kalman filter to compute estimates and error variances; plot and interpret the results.

(d) Could the FFT method have been used to compute the estimates and error variances in part (c)?

(e) Now repeat the entire problem, through part (d), but using prior kernel $\mathcal{G}_2$ rather than $\mathcal{G}_1$. Plot the two sets of estimates and error variances.

(f) How are the results from $\mathcal{G}_1$ different from those of $\mathcal{G}_2$? Pay close attention to the error variance plots.

Problem 10.4: Marching Variations

The implementation in Problem 10.3 used the standard Kalman filter. For a sufficiently large 2D or 3D domain, the regular Kalman filter might have computational limitations, and very likely issues with numerical stability. Repeat Problem 10.3(a), but instead using

(a) The square root KF of Section 4.3.1
(b) The Strip KF of Section 10.2.3
(c) The RUKF of Section 10.2.4

Problem 10.5: Acausal Marching

The implementation in Problem 10.3 used only the Kalman filter in computing estimates, which are therefore causal, whereas we know that an acausal approach, Kalman smoothing, can yield superior results. Repeat Problem 10.3(a), but using the RTS Kalman smoother, as described in Section 4.3.3.

Discuss the differences you see between the causal and acausal estimates and error variances.

Problem 10.6: Open-Ended Real-Data Problem — Large-Scale Kalman Filtering

Implement a Kalman filter for a large-scale, dynamic dataset. Many remote sensing satellites (ATSR, Topex, ERS) have data freely available online.

Avoid datasets having a complex forward problem, such as the radiometric one discussed in Application 3. Instead, we seek direct measurements, such as of ocean height (Topex) or sea-surface temperature (ATSR).

The two key challenges, as were discussed in Application 4, are that we have *sparse* measurements of a *changing* field:

- If the measurements were dense, then we would only need some sort of denoising or interpolation.
- If the underlying field were not changing, we would just have a spatial static problem with many measurements.

Develop an iterative, Kalman-filter like approach to producing dynamic estimates of the underlying random field, based on the satellite data.

11

Sampling and Monte Carlo Methods

The matter of statistical sampling was discussed in Chapter 2: *Prior Sampling* in Section 2.5.2, and *Posterior Sampling* in Section 2.5.4.

Given a random variable $\underline{z}$ obeying some prior probability density function $p(\underline{z})$, sampling from the prior distribution means generating independent random samples $\underline{z}_1, \underline{z}_2, \ldots$ from $p(\underline{z})$:

$$\underline{z}_i \sim p(\underline{z}), \tag{11.1}$$

and similarly posterior sampling from a distribution conditioned on measurements:

$$(\underline{z}_i|\underline{m}) \sim p(\underline{z}|\underline{m}). \tag{11.2}$$

The key is to decompose $(\underline{z}_i|\underline{m})$ into deterministic and stochastic components, essentially the Wold decomposition [248]:

$$(\underline{z}|\underline{m}) = (\hat{\underline{z}}|\underline{m}) + (\tilde{\underline{z}}|\underline{m}). \tag{11.3}$$

That is, the posterior $\underline{z}$ equals the estimate plus a random sample obeying the estimation error statistics, as illustrated in Figure 11.1. Much of this text has looked at methods for generating estimates $\hat{\underline{z}}$; the key to posterior sampling, then, is a means of sampling the error process $\tilde{\underline{z}}$.

The first part of this chapter develops algorithms for continuous-state prior and posterior sampling, based on the Kalman filter, marching methods, and multiscale methods, paralleling the sequence of methods presented in Chapter 10.

The substantial last part of this chapter, in Section 11.3, develops Monte Carlo and discrete-state methods. These are of key importance for discrete-state fields, in particular those as part of a hidden Markov model in Chapter 7.

The goal of Monte Carlo samplers is to find a sequence of states, dependent only on a Gibbs energy H, such that the state sequence converges to a random sample of the probability density implied by H:

P. Fieguth, *Statistical Image Processing and Multidimensional Modeling*, Information Science and Statistics, DOI 10.1007/978-1-4419-7294-1_11, © Springer Science+Business Media, LLC 2011

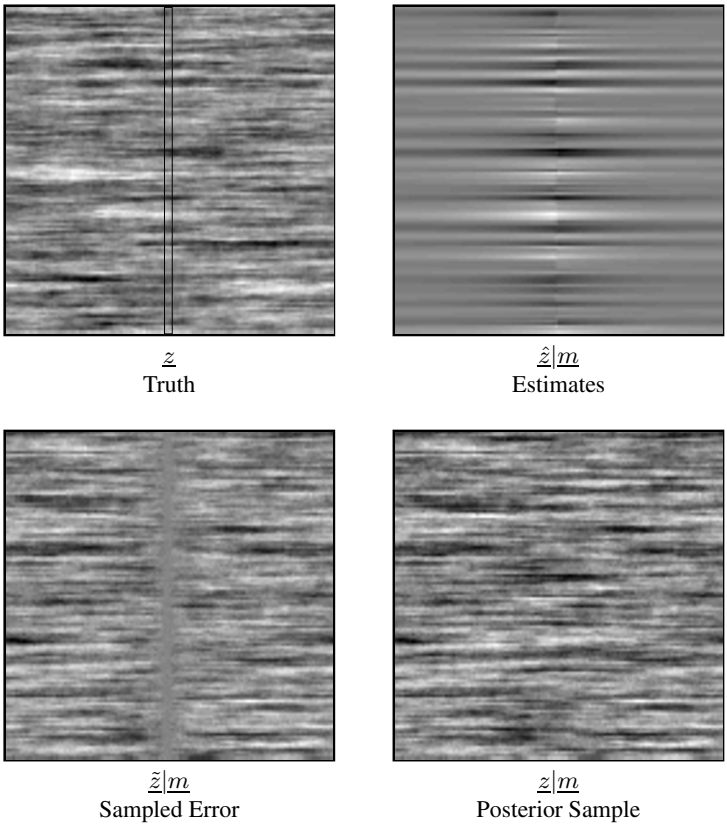


Fig. 11.1. The process of posterior sampling. The top two panels show a sample from an anisotropic prior model and estimates based on the central measured columns. The bottom-left panel shows the sampled estimation error, where a low-variance zero-mean band can be seen around the measurements, where the estimation uncertainties are small. The final panel shows the sampled posterior, consistent with both the measurements and the prior statistics. The estimates and sampled error were generated using the multiscale approach of Sections 10.3 and 11.2.3.

$$H \longrightarrow \underline{z}_1, \underline{z}_2, \ldots \qquad \text{such that} \quad \lim_{i \to \infty} \underline{z}_i \sim \frac{\exp(-\beta H)}{\mathbb{Z}}. \tag{11.4}$$

As we saw with Gibbs fields in Chapter 6 and in the context of hidden models in Chapter 7, the Gibbs model makes no particular distinction between prior and posterior models. The distinction is, in fact, only a matter of whether the measurements appear (posterior) or not (prior) in the energy function H.

11.1 Dynamic Sampling

In the context of a linear, Gauss–Markov dynamic process (4.1), the task of prior sampling is straightforward, just a simulation of the evolution of the dynamic process over time. The process initialization

$$\underline{z}(0) \sim \mathcal{N}(\underline{0}, P_0) \tag{11.5}$$

requires sampling $\underline{z}(0)$ from the prior model P_0, based on the matrix square root of P_0, as discussed in Section 2.5.2 and Appendix A.8. With the recursion initialized, dynamic prior sampling proceeds as

$$\underline{z}(t+1) = A(t)\underline{z}(t) + B(t)\underline{w}(t), \tag{11.6}$$

where $\underline{w}(t)$ is a zero-mean, unit-variance Gaussian random vector.

Next, in order to do *posterior* sampling of a dynamic process, from (11.3) it is the statistics of the estimation *error* which we need to identify, therefore for recursive posterior sampling it is really a dynamic relationship for the estimation *errors* which we require:

$$\underline{\tilde{z}}(t+1) = \tilde{A}(t)\underline{\tilde{z}}(t) + \underline{\tilde{q}}(t), \;\; \underline{\tilde{z}}(0) \sim \mathcal{N}(\underline{\tilde{z}}_o, \tilde{P}_o), \;\; \underline{\tilde{q}}(t) \sim \mathcal{N}\big(\underline{0}, \tilde{Q}(t)\big). \tag{11.7}$$

Although it is not obvious that such a form is obeyed, the error process of the Kalman filter *does* in fact obey such a dynamic process. The statistics for the predicted estimation error $\underline{\tilde{z}}(t+1|t)$ were derived in (4.124) on page 108. Rewriting for the updated estimation error $\underline{\tilde{z}}(t|t)$ yields

$$\underline{\tilde{z}}(t|t) = \underline{\hat{z}}(t|t) - \underline{z}(t) \tag{11.8}$$

$$= \big(I - K(t)C\big)A\underline{\tilde{z}}(t|t-1) + K(t)\underline{v}(t) - \big(I - K(t)C\big)B\underline{w}(t)\,; \tag{11.9}$$

consequently we can identify the dynamic parameters

$$\tilde{A}(t) = \big(I - K(t)C(t)\big)A \tag{11.10}$$

$$\tilde{Q}(t) = K(t)R(t)K^T(t) + \big(I - K(t)C(t)\big)BB^T\big(I - K(t)C(t)\big)^T \tag{11.11}$$

with initialization

$$\underline{\tilde{z}}_o = \underline{0} \tag{11.12}$$

$$\tilde{P}_o = P_o. \tag{11.13}$$

Therefore for posterior dynamic sampling we must first run the Kalman filter to compute the gain $K(t)$ for each t, then draw a random sample

$$\underline{\tilde{z}}(0) \sim \mathcal{N}(\underline{0}, P_o) \tag{11.14}$$

to initialize, then recursively compute a sample error

$$\underline{\tilde{z}}(t+1) = \tilde{A}\underline{\tilde{z}}(t) + \underline{\tilde{q}}(t) \qquad \underline{\tilde{q}}(t) \sim \mathcal{N}\big(\underline{0}, \tilde{Q}(t)\big), \tag{11.15}$$

and finally the posterior sample is computed as the sampled error added to the estimates:

$$\underline{z}|\underline{m}(t) = \underline{\hat{z}}(t) + \underline{\tilde{z}}(t). \tag{11.16}$$

The error process of the Kalman *smoother*, of Section 4.3.3, *also* obeys a dynamic process [20]. The derivation is considerably more complicated, however the recursive form of the smoothing-error process $\underline{\tilde{z}}_s$ is straightforward:

$$\begin{aligned} \underline{\tilde{z}}_s(t) &= \underline{\hat{z}}(t|T) - \underline{z}(t) && (11.17)\\ &= A^{-1}(t)\left(I - BB^T P^{-1}(t+1|t)\right)\underline{\tilde{z}}_s(t+1) - A^{-1}(t)\underline{q}_s(t), && (11.18) \end{aligned}$$

where $P(t+1|t)$ is the regular, predicted error covariance from the Kalman filter, and where the noise process $\underline{q}_s$ has covariance

$$\underline{q}_s(t) \sim B\left(I - B^T P^{-1}(t+1|t)B\right)B^T. \tag{11.19}$$

Observe that the smoothing error recursion is going backwards, from $t+1$ to t, just like the RTS smoother, which explains the presence of inverse dynamics A^{-1}.

11.2 Static Sampling

There are only few ways in which a static estimation problem may be characterized:

- Given a covariance P or inverse P^{-1}, whether prior or posterior;
- Given a set of constraints L in a regularized, non-Bayesian problem.

Let us investigate each of these two contexts in turn.

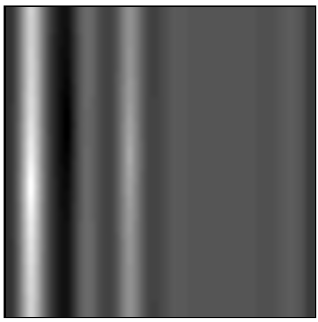
1. GIVEN COVARIANCE: Given a covariance matrix P, whether P describes a prior or posterior

$$\text{Prior:} \quad \underline{z} \sim (\underline{\mu}, P) \qquad \text{or} \qquad \text{Posterior:} \quad \underline{z}|\underline{m} \sim (\underline{\hat{z}}, P) \tag{11.20}$$

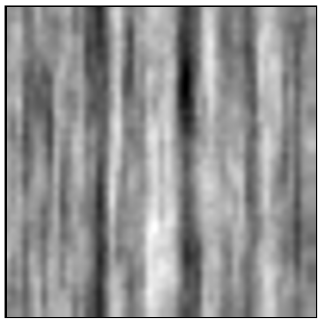
is immaterial, because in both cases the sampling process is identical:

$$\text{Let } \underline{x} \sim P \text{ be a random sample } \Rightarrow \begin{cases} \text{Prior Sampling:} & \underline{z} = \underline{\mu} + \underline{x} \\ \text{Posterior Sampling:} & \underline{z}|\underline{m} = \underline{\hat{z}} + \underline{x}. \end{cases} \tag{11.21}$$

Next, whether we are given a covariance P or its inverse P^{-1}, such as in a Markov setting, is also immaterial, since in both cases we find a matrix square root via the Cholesky decomposition (Appendix A.7.3):



Estimate $\hat{\underline{z}}$
Storage: 1.3%
Computation: 6%



Posterior Error Sample $\tilde{\underline{z}}|\underline{m}$
Storage: 12%
Computation: 50%

Fig. 11.2. An estimate and posterior sample, generated from a sparse prior, using a Cholesky decomposition. Storage and computational complexity are reported relative to a non-sparse, matrix-inversion approach. The higher storage complexity associated with the posterior sample is due to fill-in in the Cholesky step from P^{-1} to L in (11.24).

$$\begin{array}{cccccc} P & \overset{\text{Chol.}}{\longrightarrow} & P = \Gamma^T\Gamma & \longrightarrow & \underline{z} = \Gamma^T\underline{w} & \underline{w} \sim I \\ P^{-1} & \overset{\text{Chol.}}{\longrightarrow} & P^{-1} = L^TL & \longrightarrow & L\underline{z} = \underline{w} & \end{array} \tag{11.22}$$

Although the latter case appears more difficult, because L is a triangular matrix the latter $L\underline{z} = \underline{w}$ step is very simply solved by backsubstitution.

In the case where P represents the posterior covariance from an estimation

$$P = \left(C^TR^{-1}C + P_o^{-1}\right)^{-1} \tag{11.23}$$

it should be very clear that we do *not* want to explicitly calculate P as part of posterior sampling. Indeed, C and R are normally sparse or even diagonal, and in any Markov setting P_o^{-1} will be sparse-banded, therefore P^{-1} will be sparse. In (11.22), the Cholesky decomposition L will retain the sparsity of P^{-1}, making the entire sequence

$$C, R, P_o^{-1} \quad \longrightarrow \quad P^{-1} \quad \overset{\text{Chol.}}{\longrightarrow} \quad L\underline{z} = \underline{w} \tag{11.24}$$

exceptionally computationally and storage efficient, as quantified in the example in Figure 11.2.

2. GIVEN CONSTRAINTS: The constrained sampling problem is superficially straightforward, since the constraints L assert

$$L\underline{z} = \underline{w} \qquad \underline{w} \sim I \tag{11.25}$$

Type of Problem	Constraint	Rank	Random Sample
Membrane, 1D	$L = L_x$	Full-Row	$\underline{z} = L^T(LL^T)^{-1}\underline{w}$
Thin-Plate, 1D	$L = L_{xx}$	Full-Row	$\underline{z} = L^T(LL^T)^{-1}\underline{w}$
Membrane, 2D	$L = \begin{bmatrix} L_x \\ L_y \end{bmatrix}$	Rank-Deficient	$\underline{z} = L^+\underline{w}$
Thin-Plate, 2D	$L = \begin{bmatrix} L_{xx} \\ L_{yy} \end{bmatrix}$	Rank-Deficient	$\underline{z} = L^+\underline{w}$
nD Membrane + Weak Mean	$L = \begin{bmatrix} L_{mem} \\ \alpha I \end{bmatrix}$	Full-Column	$\underline{z} = (L^TL)^{-1}L^T\underline{w}$
nD Thin-Plate + Weak Mean	$L = \begin{bmatrix} L_{tp} \\ \alpha I \end{bmatrix}$	Full-Column	$\underline{z} = (L^TL)^{-1}L^T\underline{w}$

Table 11.1. Six examples of constraints, rank properties, and associated method of sampling. Whereas Bayesian approaches will always be positive-definite, and can be solved by a Cholesky decomposition, many constrained problems, such as here, will be rank-deficient and necessitate a somewhat more complex pseudoinverse.

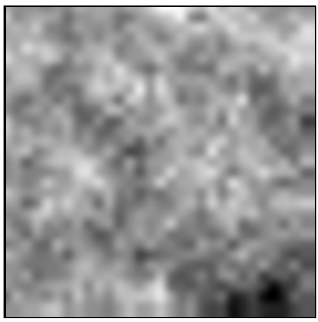
meaning that the matrix square root in (11.22) is already available in L, and all that remains is to choose a random $\underline{w}$ and to solve the linear system in (11.25).

However, whereas a given covariance P or P^{-1} is guaranteed to be positive-definite, simplifying the preceding discussion, throughout Chapter 5 we saw that the constraints matrix L is normally rectangular and may be rank-deficient.

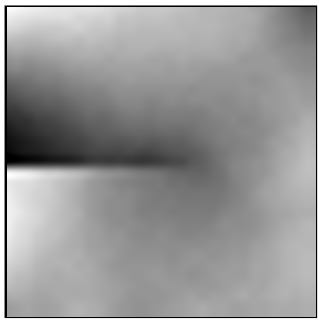
As discussed in Appendix A.9, the matrix pseudoinverse L^+ finds a solution to any linear system $L\underline{z} = \underline{w}$, returning either the unique answer if L is invertible, the least-squares answer if the problem is overconstrained, and the least-norm solution for $\underline{z}$ if the problem is underconstrained. A number of possible constraints and associated solutions are illustrated in Table 11.1, and two samples are plotted in Figure 11.3.

We need to be clear that although $\underline{z} = L^+\underline{w}$ generates a random sample, consistent with the asserted constraints $L\underline{z} = \underline{w}$, because the regularized problem is non-Bayesian and has no prior, the generated samples, such as those in Figure 11.3 cannot really be considered *prior* samples.

Finally, it should also be pointed out that in cases of full rank the pseudoinverse is easily calculated as $(L^TL)^{-1}L^T$ or $L^T(LL^T)^{-1}$, for full-column and full-row rank, respectively. As before, since LL^T and L^TL are symmetric, positive-definite, the Cholesky decomposition should be used in calculating the matrix inverse.



2D Membrane, No Boundary



2D Thin-Plate, No Boundary, Centre Cut

Fig. 11.3. Two random samples generated from rank-deficient constraints L by finding the pseudoinverse L^+. Although neither image can be considered a true prior sample, since the problems are non-Bayesian, nevertheless the samples clearly illustrate the behaviour of their respective constraints, with the cut and the nonperiodic boundaries particularly clear in the second-order example, right.

For larger problems, a nested-dissection reordering (Section 9.1.3) can be applied to improve the computational and storage efficiency of the Cholesky steps. However, to tackle sampling for very large problems we need some sort of problem transformation or domain decomposition, discussed in the following sections.

11.2.1 FFT

The FFT method of sampling was already discussed in Section 8.3, and a large fraction of the random-field images plotted in this text were generated using the FFT approach.

For fully stationary problems with periodic boundary conditions, the FFT diagonalizes the problem, allowing problems of nearly arbitrary size to be tackled. The prior sample

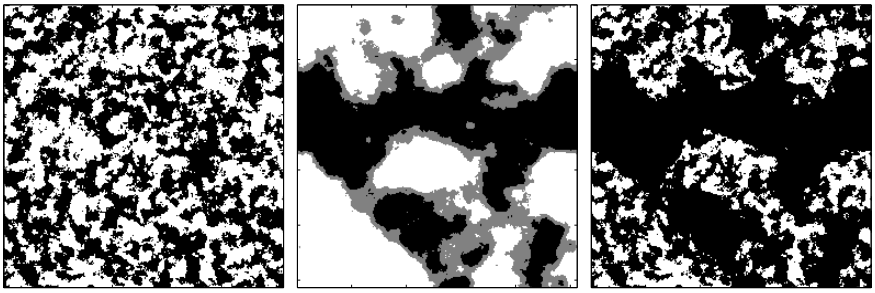
$$Z = \text{FFT}_d^{-1}\left(\sqrt{\text{FFT}_d(\mathcal{P})} \odot \text{FFT}_d(W)\right) \tag{11.26}$$

and posterior sample

$$(Z|M) = \hat{Z} + \text{FFT}_d^{-1}\left(\sqrt{\text{FFT}_d(\tilde{\mathcal{P}})} \odot \text{FFT}_d(W)\right) \tag{11.27}$$

were developed in (8.78),(8.82).

Even if a problem is not fully stationary, the FFT approach may still have merit in generating samples of stationary portions of a problem, or in generating stationary samples ignoring the nonstationary aspects of a boundary, for example.



$\text{sign}(Z_f)$
Binary Sampling

$\text{sign}(Z_c - 2) + \text{sign}(Z_c + 2)$
Ternary Sampling

$\text{sign}(Z_c) \odot \text{sign}(Z_f)$
Two-Scale Sampling

Fig. 11.4. Given one or more continuous-state FFT samples, discretizing them gives an *ad hoc*, but very fast, approach to discrete-state sampling. By combining multiple fields, right, samples with relatively complex, multiscale morphology can be synthesized. Here, Z_f and Z_c are continuous-state thin-plate samples at fine and coarse scales, respectively.

Although most discrete-state sampling methods are based on the MCMC methods in Section 11.3, a FFT sample can be discretized [208] as a very fast, albeit heuristic, approach. Three examples are shown in Figure 11.4; clearly quite creative things can be done by combining multiple random fields, and the efficiency of the FFT allows this approach to be generalized to three-dimensional domains.

11.2.2 Marching

In Section 10.1 we took advantage of the static-dynamic duality, originally discussed in Section 4.1.2, to allow a static problem to be partitioned and estimated recursively using a Kalman filter. Clearly precisely the same concept can be extended from estimation to sampling.

That is, given the statistics of the static problem and a proposed partitioning, we can identify the cross-statistics to form a dynamic model, as described in Section 10.1.

For prior sampling, we apply the identified dynamic model to the dynamic sampler of Section 11.1. In the case of posterior sampling we run the Kalman filter, possibly large (as in Section 10.2), build the dynamic model (11.7) for the estimation errors, and compute the posterior samples from (11.15).

The difficulty in sampling from a marching approach is that the marching approach is causal, whereas most static problems are noncausal, as was illustrated in Example 10.1. Consequently we would normally prefer smoothing, as discussed in Section 10.2.1.

The simple, *ad hoc* approach to smoothing illustrated in Example 10.1, where causal and anticausal estimates are combined, cannot apply to random prior or posterior sampling, however. The reason is that multiple suboptimal estimates $\hat{\underline{z}}_i$ are all attempting to estimate the *same* quantity $\hat{\underline{z}}$, therefore the $\{\hat{\underline{z}}_i\}$ are correlated and can be averaged:

$$\hat{\underline{z}}_i \sim (\hat{\underline{z}}, \tilde{P}_i) \quad \Longrightarrow \quad E_i\big[\hat{\underline{z}}_i\big] \approx \hat{\underline{z}}. \tag{11.28}$$

On the other hand, *random* samples $\tilde{\underline{z}}_i$ are driven by independent noise processes, and do *not* average meaningfully:

$$\tilde{\underline{z}}_i \sim (\underline{0}, \tilde{P}_i) \quad \Longrightarrow \quad E_i\big[\tilde{\underline{z}}_i\big] \approx \underline{0}. \tag{11.29}$$

It is for precisely the same reason that the overlapped approach of Section 8.2.3, which performs pointwise averaging simlar to (11.28)–(11.29), is much more effective for estimation than for sampling.

In contrast, if a sparse Kalman filter is implemented such that it is possible to save the covariance information in order to implement the regular Kalman smoother, then the recursive form of the smoothing-error process (11.18) can be used to generate posterior samples.

11.2.3 Multiscale Sampling

The multiscale method of Section 10.3 obeys a scale-to-scale dynamic model (10.54),

$$\underline{x}(s) = A(s)\underline{x}\big(\uparrow s\big) + B(s)\underline{w}(s), \tag{11.30}$$

so prior sampling in the multiscale environment follows trivially from Section 11.1. The linearity of the multiscale model allows the prior mean to be separated from the remainder of the problem, so we can, without loss of generality, assume zero mean.

We begin at the root node

$$P(0) \xrightarrow{\text{Chol.}} P(0) = \Gamma^T\Gamma \quad \Rightarrow \quad \underline{x}(0) = \Gamma^T\underline{q} \qquad \underline{q} \sim I. \tag{11.31}$$

After this one Cholesky decomposition, the remaining nodes follow per the dynamics:

$$\underline{x}(s) = A(s)\underline{x}\big(\uparrow s\big) + B(s)\underline{w}(s) \qquad \underline{w}(s) \sim I \tag{11.32}$$

Next, since the multiscale estimator was derived from the RTS smoother, and that a smoothing-error process (11.18) has been derived [20] for the RTS smoother, it turns out to be true that the error in the estimates of the multiscale method *itself* obeys a multiscale model [215]:

$$\tilde{\underline{x}}(s) = \tilde{A}(s)\tilde{\underline{x}}\big(\uparrow s\big) + \underline{\tilde{w}}(s), \;\; \underline{\tilde{w}}(s) \sim \mathcal{N}\big(\underline{0}, \tilde{Q}(s)\big), \;\; \tilde{\underline{x}}(0) \sim \mathcal{N}(\hat{\underline{x}}(0), \tilde{P}_0), \tag{11.33}$$

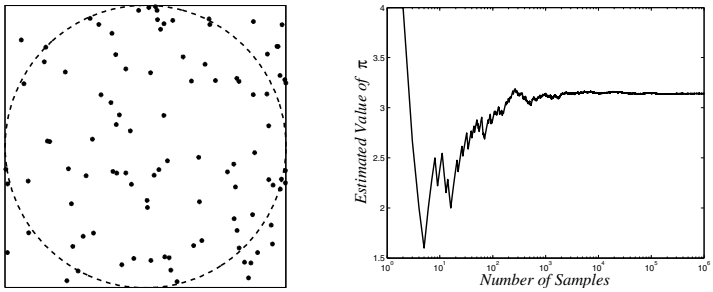


Fig. 11.5. A random sampling approach to estimating the value of π. In the left panel, the area of the square is 4, and the area of the dashed circle is $\pi \cdot 1^2$. Therefore, placing uniformly-distributed random samples and multiplying the fraction inside the circle by four gives us an estimate of the value of π, right.

where $\tilde{A}(s), \tilde{Q}(s)$ are expressed [215] in terms of the model parameters $A(s), B(s)$, prior covariances $P(s)$, and estimation error covariances $\tilde{P}_u(s)$ computed in the upwards pass of the multiscale estimator:

$$\tilde{A}(s) = \tilde{P}_u(s)F^T(s)\tilde{P}_u^{-1}(\uparrow s) \tag{11.34}$$

$$\tilde{Q}(s) = \tilde{P}_u(s) - \tilde{P}_u(s)F^T(s)\tilde{P}_u^{-1}(\uparrow s)F(s)\tilde{P}_u(s) \tag{11.35}$$

$$F(s) = P(\uparrow s)A^T(s)P^{-1}(s). \tag{11.36}$$

The availability of a multiscale smoothing error model leads to an approach to hierarchical posterior sampling, following the dynamic approach of Section 11.1:

1. The multiscale estimator computes $\underline{\hat{x}}(s), \tilde{P}(s)$ at each node on the tree.
2. Infer the smoothing error model $\tilde{A}(s), \tilde{Q}(s)$ from (11.34)–(11.36).
3. Using the Cholesky decomposition compute the matrix square roots

$$\tilde{\Gamma}(0) = \left(\tilde{P}(0)\right)^{1/2} \qquad \tilde{\Gamma}(s) = \left(\tilde{Q}(s)\right)^{1/2}, s \neq 0. \tag{11.37}$$

4. Sample the smoothing error process, coarse to fine:

$$(\underline{\tilde{x}}|\underline{m})(0) = \tilde{\Gamma}^T(0)\underline{\tilde{w}}(0), \qquad \underline{\tilde{w}}(0) \sim \mathcal{N}(\underline{0}, I) \tag{11.38}$$

$$(\underline{\tilde{x}}|\underline{m})(s) = \tilde{A}(\underline{\tilde{x}}|\underline{m})\big(\uparrow s\big) + \tilde{\Gamma}^T(s)\underline{\tilde{w}}(s), \qquad \underline{\tilde{w}}(s) \sim \mathcal{N}(\underline{0}, I). \tag{11.39}$$

5. Finally, add the mean (the estimates) to produce the posterior sample:

$$(\underline{x}|\underline{m})(s) = \underline{\hat{x}}(s) + (\underline{\tilde{x}}|\underline{m})(s). \tag{11.40}$$

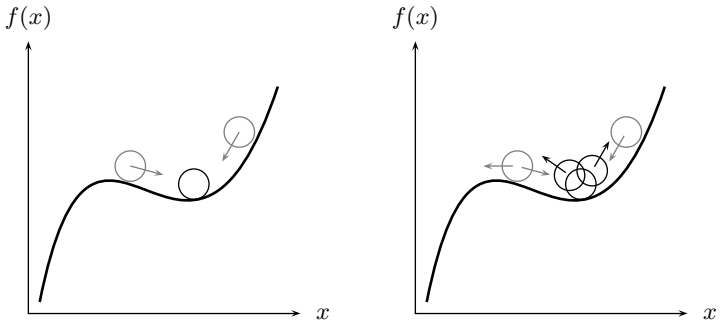


Fig. 11.6. Regular gradient descent, left, will converge to and stay in a local minimum. Stochastic gradient descent, right, attempts to escape local minima by accepting *inferior* values of x with some probability, essentially perturbing or agitating the current position in order to overcome barriers of limited height.

If the state $\underline{z}(s)$ has dimension $n(s)$, then each of the estimation, smoothing, and square root steps has complexity $\mathcal{O}(n(s)^3)$ per node.

The posterior sampling images in Figure 11.1 were calculated via posterior sampling on a multiscale model.

11.3 MCMC

The great increase in computer processing power over the last twenty years has created opportunities for a class of algorithms, exceptionally simple in concept, but with very large computational complexity. The methods, known collectively as Markov Chain Monte Carlo (MCMC) methods [42, 64, 88, 137, 155, 194, 335], are based on using randomness to solve mathematical goals.

A simple, classic example is shown in Figure 11.5, whereby random samples can be used to estimate the value of π. Although the same approach could be used with a non-random dense, regular grid, such a grid requires the number of sample points to be decided upon ahead of time, and a valid answer is reached only when the grid is complete.

The MCMC approach is impractical for the specific problem of estimating π, since there exist much faster specialized methods. Nevertheless the random sampling approach is an exceptionally elegant and simple idea, capable of sampling from *any*

Algorithm 10 Single Gibbs Sample

Goals: Perform Gibbs sampling at site i in $\underline{z}$ based on $H()$
Function $\hat{\underline{z}} = \textbf{SingleGibbs}(H(), T, \underline{z}, i, \Psi)$
  $\hat{\underline{z}} \leftarrow \underline{z}$
  for $j \in \Psi$ **do**
    $h(j) \leftarrow \textbf{exp}\big(H(\underline{z}, \underline{z}_i = j)/T\big)$ *Test all possible state values.*
  end for
  $s \leftarrow \textbf{sum}(\underline{h})$ *Compute marginal partition function*
  $r \leftarrow \textbf{UniformRandom}()$
  $p \leftarrow 0$
  for $j \in \Psi$ **do**
    $p \leftarrow p + h(j)/s$
    if $r \leq p$ **then**
      $\hat{\underline{z}}_i \leftarrow j$ *Sample state j from marginal distribution.*
      Return
    end if
  end for

distribution, with further generalizations to sampling state spaces of variable dimensions [147].

Consider stochastic gradient descent, as illustrated in Figure 11.6. Regular gradient descent seeks to minimize an objective function $f(x)$ by moving against the gradient,

$$x_{k+1} - x_k - \alpha f'(x_k), \tag{11.41}$$

but may converge to, and stay in, a local minimum, where $f'(x) = 0$. Stochastic gradient descent adds a degree of random variability, such that the current point has a nonzero probability of accepting an inferior solution by moving "uphill," increasing the likelihood of escaping local minima. If we let T_k be the "temperature" parameter (the degree of agitation) at iteration k, then T_k needs to decrease to zero to allow the system to finally converge:

$$\begin{array}{lcl} \text{Rapid decrease in } T_k & \rightarrow & \text{Fast convergence, more likely stuck in local minimum} \\ \text{Slow decrease in } T_k & \rightarrow & \text{Slower convergence, more likely at global minimum.} \end{array} \tag{11.42}$$

This concept and resulting tradeoff is the essence of *annealing*, discussed in the following section.

11.3.1 Stochastic Sampling

Rather than random perturbations of a ball on a hill, we are interested in considering random state transitions $Z \Rightarrow \bar{Z}$ with transition probability $P(Z \Rightarrow \bar{Z})$. Indeed, consider a sequence of perturbations

$$Z_1 \rightarrow Z_2 \rightarrow \cdots \tag{11.43}$$

Algorithm 11 Single Metropolis Sample

Goals: Consider a state change $\underline{z} \rightarrow \bar{\underline{z}}$ on the basis of the probability density implied by $H()$
Function $\hat{\underline{z}} = \textbf{SingleMetropolis}(H(), T, \underline{z}, \bar{\underline{z}})$
  $h \leftarrow H(\underline{z})/T$
  $\bar{h} \leftarrow H(\bar{\underline{z}})/T$
  $r \leftarrow \textbf{UniformRandom}()$
  if $r < \textbf{exp}(h - \bar{h})$ **then**
    $\hat{\underline{z}} \leftarrow \bar{\underline{z}}$
  else
    $\hat{\underline{z}} \leftarrow \underline{z}$
  end if

Remarkably, this sequence will converge [144, 335] to a random sample of distribution $p(Z)$ if two conditions hold:

IRREDUCIBILITY: Every state must be reachable,

$$P^{(k)}(Z \Rightarrow \bar{Z}) > 0 \quad \forall\, Z, \bar{Z} \tag{11.44}$$

for some $k > 0$, where $P^{(k)}$ is the k-step transition probability.

DETAILED BALANCE: There is a balanced equilibrium,

$$p(Z)P(Z \Rightarrow \bar{Z}) \equiv p(\bar{Z})P(\bar{Z} \Rightarrow Z). \tag{11.45}$$

The two most common choices of state perturbation or transition are the Gibbs [335] and Metropolis samplers:

GIBBS: (Algorithm 10)
Randomly sample a single state element Z_i from its marginal distribution:

$$P(Z_i \rightarrow \alpha | Z) = \frac{p(Z \backslash Z_i = \alpha)}{\sum_{\beta \in \Psi} p(Z \backslash Z_i = \beta)} \tag{11.46}$$

METROPOLIS: (Algorithm 11)
The transition $Z \Rightarrow \bar{Z}$ depends on the relative likelihoods:

$$\begin{array}{l} p(\bar{Z}) > p(Z) \text{ Accept transition with probability 1.} \\ p(\bar{Z}) \leq p(Z) \text{ Accept transition with probability } \frac{p(\bar{Z})}{p(Z)} \end{array} \tag{11.47}$$

That the Metropolis transition satisfies detailed balance (11.45) is obvious. Supposing, without loss of generality, that $p(Z) > p(\bar{Z})$; then

$$\begin{array}{ccc} \underbrace{p(Z)\underbrace{P(Z \Rightarrow \bar{Z})}_{\frac{p(\bar{Z})}{p(Z)}}}_{p(\bar{Z})} & \overset{?}{\equiv} & \underbrace{p(\bar{Z})\underbrace{P(\bar{Z} \Rightarrow Z)}_{1}}_{p(\bar{Z})} \\ & \overset{\checkmark}{=} & \end{array} \tag{11.48}$$

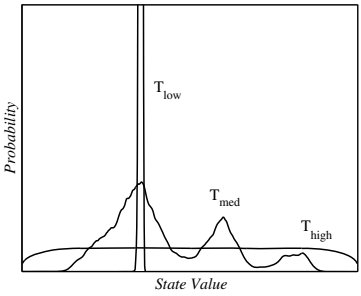


Fig. 11.7. As temperature T is decreased, the probability density $\exp(-H/T)/\mathbb{Z}(T)$ is weighted more and more heavily towards peaks in the distribution.

Recalling the definition of a Gibbs distribution[1] from (6.39),

$$p(\underline{z}) = \frac{1}{\mathbb{Z}} e^{-H(\underline{z})/T}. \tag{11.49}$$

what is significant is that neither sampler requires the calculation of the difficult partition function $\mathbb{Z}$, since the Gibbs sampler needs only *marginal* distributions, and Metropolis needs only *ratios* of distributions in which $\mathbb{Z}$ cancels:

$$\frac{p(\bar{Z})}{p(Z)} = \frac{\frac{1}{\mathbb{Z}} e^{-H(\bar{Z})/T}}{\frac{1}{\mathbb{Z}} e^{-H(Z)/T}} = \exp\left(\frac{H(Z) - H(\bar{Z})}{T}\right). \tag{11.50}$$

Furthermore, recalling from (6.42) that H is a sum of potential functions over cliques,

$$H(Z) = \sum_{c \in \mathcal{C}} V\big(\{z_i, i \in c\}\big), \tag{11.51}$$

and letting

$$\bar{\mathcal{C}} = \{c \,|\, Z_c \neq \bar{Z}_c\} \tag{11.52}$$

be the clique set over which $Z, \bar{Z}$ are not identical, then the probability ratio from (11.50) becomes

$$\frac{p(\bar{Z})}{p(Z)} = \exp\left(\frac{\sum_{c \in \bar{\mathcal{C}}} V_c(Z_c) - V_c(\bar{Z}_c)}{T}\right), \tag{11.53}$$

which is a straightforward sum over a modest number of potential functions, with no partition function or probability density!

[1] Strictly speaking, in (6.39) Gibbs parameter $\beta = 1/kT$, where k is Boltzmann's constant. Since we are not working with actual physics-based Hamiltonians H, constant k is unimportant and we drop it for clarity.

Algorithm 12 Basic, Single-Scale Annealing

Goals: Solve the optimization problem $\hat{\underline{z}} = \arg_{\underline{z}} \min H(\underline{z})$ with annealing schedule T_k
Function $\hat{\underline{z}} = \textbf{Anneal}(H(), \{T_k\}, \underline{z}_0)$
  $k \leftarrow 0$
  while not converged **do**
    $\underline{z}_{k+1} \leftarrow \textbf{MetropolisSampler}(\underline{z}_k, H(), T_k)$
    $k \leftarrow k + 1$
  end while
  $\hat{\underline{z}} \leftarrow \hat{\underline{z}}_k$

Next, following up on the illustration in Figure 11.6, what role does "temperature" T play in (11.53)? For a fixed value of T, we are performing random sampling, either posterior or prior — depending on whether H includes measurement terms. If T is decreased with iteration, we are slowly "cooling" the random perturbations, a process known as *simulated annealing* [126, 127, 335], shown in Algorithm 12. As $T \rightarrow 0$ we are sampling from the global minimum (or possibly multiple global minima) of H, as illustrated in Figure 11.7:

$$\lim_{T \rightarrow 0} p(Z) = \lim_{T \rightarrow 0} \frac{\exp\bigl(-H(Z)/T\bigr)}{\mathbb{Z}(T)} = \frac{1}{|\mathcal{Z}_{\text{min}}|} \sum_i \delta(Z - Z_{\text{min}}^{(i)}), \tag{11.54}$$

where

$$Z_{\text{min}}^{(i)} \in \mathcal{Z}_{\text{min}} = \bigl\{\arg_Z \min H(Z)\bigr\} \equiv \bigl\{\arg_Z \max p(Z)\bigr\}. \tag{11.55}$$

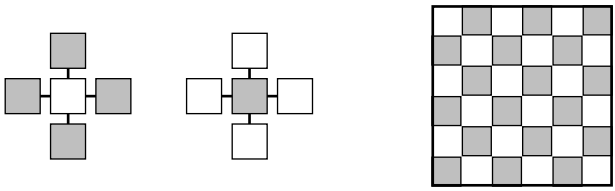
If H contains measurement terms, then $p()$ is a posterior probability, in which case the annealed sample is taken from the set of points maximizing the posterior distribution, meaning that we have found an MAP estimate:

$$Z\Big|_{T \rightarrow 0} = \hat{Z}_{MAP}(M). \tag{11.56}$$

As already discussed in (11.42), whether the sampler actually converges to this global optimum is a function of the temperature schedule T_k. Theoretically [127], to converge globally requires an exceptionally slow logarithmic schedule $T_k \propto 1/\log(k)$. In practice, most approaches choose to cool much more quickly, known as *quenching*, commonly using an exponential schedule $T_k \propto \exp(-kr)$ for some cooling rate r.

11.3.2 Continuous-State Sampling

The previous section was deliberately ambiguous as to whether the state Z was a discrete- or continuous-state process. Although uncommon, it is possible to sample and anneal continuous-state fields.



Two different
first-order neighbourhoods

A checkerboard pattern of
non-interacting state sets

Fig. 11.8. A first-order model, left, on a two-dimensional regular grid consists of two non-interacting sets (light and dark) of state elements, arranged as a checkerboard pattern. Because of the non-interaction, all of the elements in each set can be sampled in parallel.

Because the Gibbs sampler (Algorithm 10) samples from the marginal distribution, in the continuous-state case this would require the difficult numerical characterization of a continuous-state PDF [176]. In contrast, the Metropolis sampler (Algorithm 11) requires only the evaluation of the relative energies at the current $H(Z)$ and proposed $H(\bar{Z})$ states, however there remains a challenge of generating meaningful proposals, since in the infinity of possible continuous-state values, only an infinitesimal fraction represent meaningful alternatives.

As we have seen throughout this text, there are many methods for solving regularly-gridded continuous-state problems, and sampling / annealing methods have little to offer. However there are unstructured parametric problems, known as marked point processes [246, 292], where continuous-state annealing has provided promising results. In these problems, a relatively small number of continuous values parametrize the behaviour of an image, such as the locations of roads [292] or the orientations and sizes of buildings [245].

11.3.3 Large-Scale Discrete-State Sampling

The primary application of MCMC methods in the context of this text is to large-scale discrete-state fields, found particularly in the hidden fields of Chapter 7, even more specifically as the estimator in the E-Step of the EM algorithm used with hidden fields, as described in Section 7.5.

In the somewhat rare event that our discrete-state field is acyclic (having no loops; see Figure 5.1 on page 135), whether sequential or tree-based, then there are very efficient approaches for producing state estimates based on the Viterbi algorithm of Section 4.5.1: the forward–backward algorithm [265, 275] for sequential problems, and a very similar upward–downward algorithm on trees [77, 275].

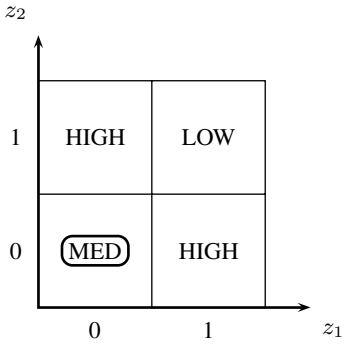


Fig. 11.9. The energy map for the simple, two-state problem of (11.58) is shown. The only likely states are $\underline{z}^T = [0\ 0]$ and $\underline{z}^T = [1\ 1]$. If the current state is $\underline{z}^T = [0\ 0]$, then it is not possible to reach $[1\ 1]$ in a single-state flip, however the intermediate states $[0\ 1]$, $[1\ 0]$ are very improbable, so there is a significant barrier in place to the $[0\ 0] \rightarrow [1\ 1]$ transition unless both states change simultaneously.

It is much more common, however, for the discrete states to live on a multidimensional regular grid. As with the marching methods of Chapter 10, we can choose to impose a sequential ordering of the grid elements (using a *Peano* or space-filling curve), in which case the efficient forward–backward algorithm applies, however such an ordering brings the same modelling drawbacks as we saw with the causal random fields and symmetric half-plane models of Chapter 6.

In general, given a multidimensional discrete-state problem the most immediate acceleration is vectorization, processing non-interacting state elements simultaneously. Specifically, given a set S of non-interacting sites

$$S = \{s_1, s_2, \ldots\} \qquad \text{such that} \qquad s_i \notin \mathcal{N}_{s_j} \quad \forall\, i, j \tag{11.57}$$

then, if the neighbourhood $\mathcal{N}$ is relatively small in extent, $|S|$ will be fairly large. Best known is the checkerboard vectorization, for first-order models such as Ising, whereby the entire grid can be divided into only two sets S_{black} and S_{white}, as illustrated in Figure 11.8.

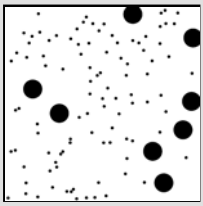
Methods of acceleration beyond vectorization must somehow address the problem of state coupling, the problem of indirection, and the random walk phenomenon of Figure 8.18 in Section 8.6, whereby the number of iterations for convergence is a quadratic function of structure size or correlation length.

A related problem is the concept of an *energy barrier*. Suppose we have a simple, two-state Ising model for which the energy map is plotted in Figure 11.9:

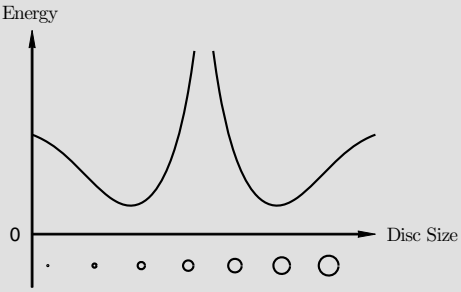
$$H(\underline{z}) = -\beta\delta_{z_1,z_2} - \alpha z_1 \qquad z_1, z_2 \in \{0, 1\} \quad \beta \gg \alpha \gg 0 \tag{11.58}$$

Example 11.1: Annealing and Energy Barriers

Suppose we have a large two-dimensional field consisting of small and large discs:

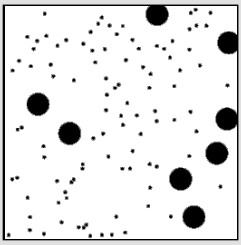


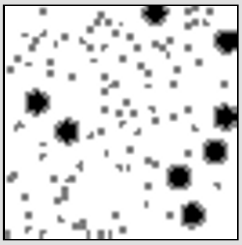
An energy function corresponding to such a field will wish to encourage small and large discs, but to penalize discs of other radii,

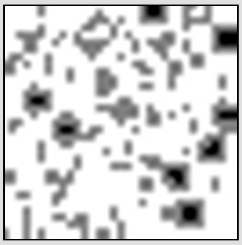


creating an energy barrier. If an annealer were initialized with a random set of pixel values, it would quickly reduce the energy (increase the probability) by clumping the random values into small discs, however the energy function will make it very difficult to anneal small discs gradually into larger ones, because of the strong penalty placed on intermediate sizes.

We can try to avoid this energy barrier by formulating the problem at coarser scales, where incremental changes correspond to the simultaneous changing of *many* state elements at the finest scale:


Down 2 Scales


Down 4 Scales


Down 5 Scales

At the coarsest scale, only two to four "black" elements are required in order to produce a large, finest-scale "black" disc of approximately 7000 pixels.

Algorithm 13 Multilevel Annealing, Initializing from Coarser Scale

Goals: Solve $\hat{\underline{z}} = \arg_{\underline{z}} \min H(\underline{z})$ by annealing, each scale initialized by the preceding

Function $\hat{\underline{z}} =$ **HierAnneal**$(Scales, \{H^j()\}, \{T_k^j\})$
  $\hat{\underline{z}}^{Scales} \leftarrow$ **Anneal**$(H^{Scales}, \{T_k^{Scales}\})$ *Regularly anneal coarsest scale*
  for $j \leftarrow Scales - 1$ **to** 0 **do**
    $\underline{z}_0^j \leftarrow$ **Project**$(\hat{\underline{z}}^{j+1})$ *Project to finer scale*
    $\hat{\underline{z}}^j \leftarrow$ **Anneal**$(H^j(\cdot), \{T_k^j\}, \underline{z}_0^j)$ *Anneal at scale j*
  end for

Starting in state $[0\ 0]$, to get to the optimum state $[1\ 1]$ with single-site state changes we need to pass through highly unlikely intermediate states $[0\ 1]$ or $[1\ 0]$, an energy barrier. Some sort of simultaneous or multiple-state flipping is required.

In general, there are two possibilities for reducing computational complexity:

1. Reduce the structure size or correlation length by downsampling; that is, by creating a tree or hierarchy. Examples include multigrid Monte Carlo [144], and discrete-state hierarchies [5, 50, 186, 197, 234]. Two illustrations of such a hierarchy were shown in Chapter 8 in Figure 8.22 (page 285).

2. Reduce the structure size or correlation length by making state elements larger; that is, by grouping. Examples include Swendsen–Wang [184, 297] and region grouping [217, 314, 329]; both examples were illustrated Figure 8.21 (page 284).

The latter approach fits naturally with the Metropolis sampler, since in a proposed change $Z \rightarrow \bar{Z}$ there is no implication that only *one* state element changes, rather it is perfectly reasonable to have a whole group or clump of state elements change. In particular, in (11.58) and Figure 11.9, the Metropolis state change $[0\ 0] \rightarrow [1\ 1]$ would immediately be accepted. However, exactly analogous to the Metropolis sampling in the continuous-state case in Section 11.3.2, there again remains a challenge of generating meaningful proposed transitions $\bar{Z} \rightarrow Z$, since only an infinitesimal fraction of flipped state groupings actually represent likely alternatives.

The former approach, constructing the problem on multiple scales, involves learning energy functions H^j as a function of scale j, but is ultimately more straightforward than grouped Metropolis since a regular Gibbs sampler can be used. Two general approaches to hierarchical annealing may be developed, where the distinction lies in how the coarser scale affects the finer one:

- As the *initialization* to the finer scale, as outlined in Algorithm 13, and sketched in the top sequence in Figure 8.22;

- As a *constraint* on the energy function of the finer scale, corresponding to Algorithm 14, and shown in the bottom sequence in Figure 8.22.

Algorithm 14 Multilevel Annealing, Constrained by Coarser Scale

Goals: Solve $\hat{\underline{z}} = \arg_{\underline{z}} \min H(\underline{z})$ by annealing, each scale constrained by the preceding

Function $\hat{\underline{z}} = \textbf{HierAnneal}(Scales, \{H^j()\}, \{T_k^j\})$
  $\hat{\underline{z}}^{Scales} \leftarrow \textbf{Anneal}(H^{Scales}, \{T_k^{Scales}\})$ *Regularly anneal coarsest scale*
  for $j \leftarrow Scales - 1$ **to** 0 **do**
    Randomly initialize $\underline{z}_0^j$
    $\hat{\underline{z}}^j \leftarrow \textbf{Anneal}(H^j(\cdot|\hat{\underline{z}}^{j+1}), \{T_k^j\}, \underline{z}_0^j)$ *Anneal at scale j*
  end for

11.4 Nonparametric Sampling

Most of this text has focused on model-based methods, where an *explicit* model is learned.[2] However, where the behaviour of a random field is difficult to parametrize, it may be possible to model the field *implicitly* in terms of the given training data.

Recall from (2.4) (page 15) that an inverse problem could be solved, in principle, as

$$\underline{z} = f^{-1}(\underline{m}) \triangleq \{\underline{z} | f(\underline{z}) = \underline{m}\}, \tag{11.59}$$

however the number of possible configurations $\{\underline{z}\}$ is huge and computationally infeasible to enumerate. Given a number of training samples $\check{\underline{z}}_i$, we can try to solve the inverse problem with an implicit prior model by limiting to the training samples,

$$\hat{\underline{z}} = \check{\underline{z}}_i \quad \text{such that } f(\check{\underline{z}}_i) = \underline{m}, \tag{11.60}$$

however it is inconceivable that the correct solution to the inverse problem would appear, perfectly, in a finite training sample.

We have two difficulties: the number of possible configurations $\{\underline{z}\}$, and the problem of *existence*, whether a $\underline{z}$ even exists such that $f(\underline{z}) = \underline{m}$. We can address both of these problems by dividing $\underline{z}$ into small patches, similar to the local reduction of basis in Section 8.2.3, and by using a least-squares (or other) norm in assessing the fit between a local patch and its corresponding measurements.

Indeed, a wide variety of patch-based sampling methods has recently been developed [94, 119, 209].

Let us begin with the simpler problem of prior sampling, most commonly applied to texture synthesis [94, 209]. Suppose we wish to synthesize a sample, one by one constructing the pixels of Z from $\check{Z}$,

$$\check{Z}_i \in \mathbb{R} \qquad Z_i \in \{\mathbb{R}, \text{NaN}\}, \tag{11.61}$$

where "NaN" (not a number) allows us to distinguish between asserted and undetermined elements in Z. We wish to examine Z and $\check{Z}$ patchwise, so let

[2] With the notable exception of conditional random fields in Section 7.3.

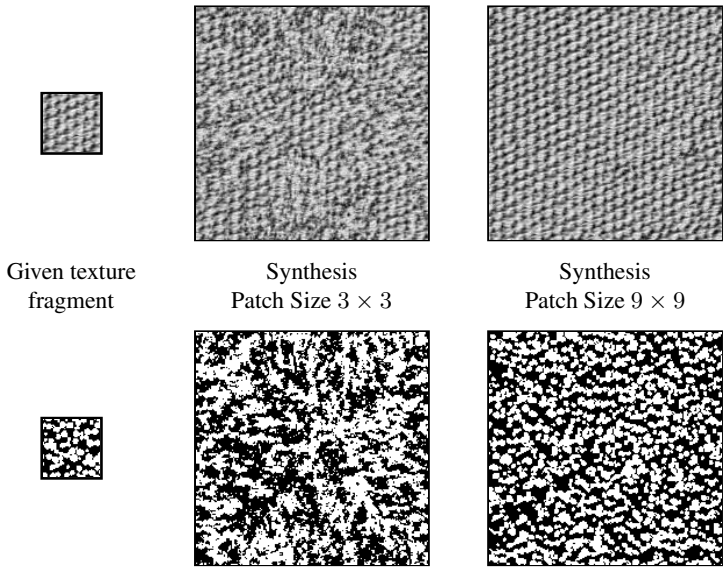


Fig. 11.10. Nonparametric prior sampling: Given a ground-truth image, left, random samples can be generated by matching patches between ground-truth and sample. Aspects of pattern "memorization" or copying can be seen in the bottom-right synthesis.

$$\mathcal{N}_i, \quad i \in \mathcal{N}_i \tag{11.62}$$

be the region or neighbourhood surrounding location i. We can therefore test the goodness-of-fit between some partial patch Z and any part of $\check{Z}$:

$$\left\| \check{Z}_{\mathcal{N}_j}, Z_{\mathcal{N}_i} \right\|. \tag{11.63}$$

With a norm in place, we wish to randomly sample a pixel Z_i on the basis of the patches in $\check{Z}$ which match the region in Z surrounding i:

$$Z_i \sim \left\{ \check{Z}_j | \left\| \check{Z}_{\mathcal{N}_j}, Z_{\mathcal{N}_i} \right\| = 0 \right\}. \tag{11.64}$$

Because $\check{Z}$ is of finite size we cannot expect a perfect match, so the norm criterion needs to be moderated:

$$Z_i \sim \left\{ \check{Z}_j | \left\| \check{Z}_{\mathcal{N}_j}, Z_{\mathcal{N}_i} \right\| \leq \epsilon \right\}. \tag{11.65}$$

Z is initialized with a tiny patch of pixels from $\check{Z}$, after which (11.65) is repeatedly applied until the entire domain is sampled. The elegance of the approach is that only ϵ and the patch size $|\mathcal{N}|$ need to be specified; all other aspects of the model are nonparametric.

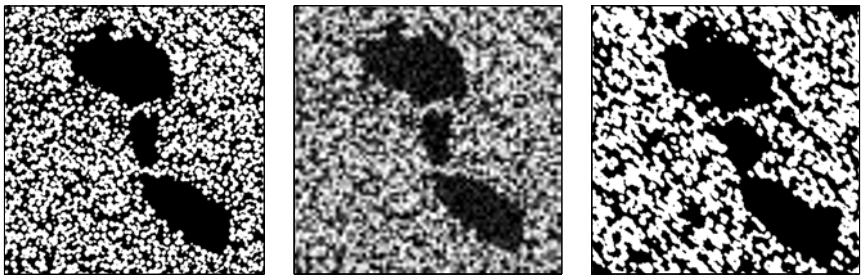


Fig. 11.11. Patch-based Superresolution: It is possible to infer a high-resolution image Z, right, from low-resolution measurements, middle, given high-resolution data $\check{Z}$ which constrain the permissible patches in Z.

Two illustrations are shown in Figure 11.10. The synthesized patterns are quite credible, however there is a clear dependence on patch size. Sizes which are too small (local) fail to adequately sense the spatial texture, whereas sizes which are too large will tend to "memorize" $\check{Z}$, such that only a single patch matches in (11.65), and the synthesized pattern grows nearly deterministically.

Generalizing the above patch-based method to solving inverse problems and posterior sampling may be difficult:

1. We cannot assume a dense, regular grid for the measurements $\underline{m}$.
2. The forward model $f()$ may be nonlocal, a poor fit to the local, nonparametric model of the patch methods.

There is, however, an important class of problems involving resolution-enhancement or *superresolution* [14, 104], in which we wish to find a high-resolution image based on a ground-truth image and low-resolution measurements (for example see Application 11, following this section). Because the low-resolution measurements are dense and downsampling is a local operator, the above two difficulties do not apply, and nonparametric patch methods have been developed [90, 119] for this case.

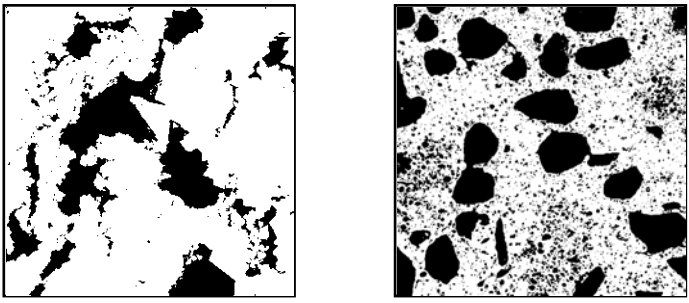
Although the implementation details are somewhat complicated, essentially we now seek a pixel from those patches simultaneously satisfying the forward problem and the already-sampled parts of Z:

$$Z_i \sim \left\{ \check{Z}_j \middle| \left\| \check{Z}_{\mathcal{N}_j}, Z_{\mathcal{N}_i} \right\| \leq \epsilon, \ \left\| \underline{m}_i, f(\check{Z}_{\mathcal{N}_j}) \right\| \leq \delta \right\}, \tag{11.66}$$

where $\underline{m}_i$ is the subset of the measurements which apply to region $\mathcal{N}_i$ around pixel location i. An example illustrating example-based superresolution is shown in Figure 11.11.

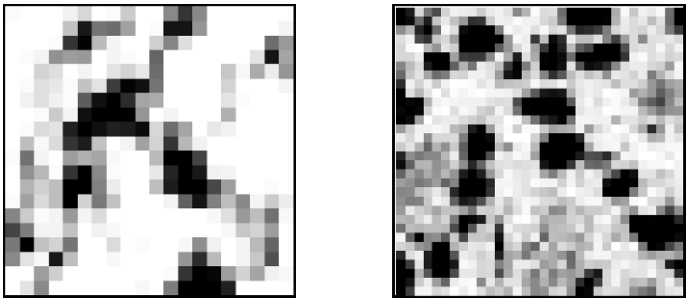
Application 11: Multi-Instrument Fusion of Porous Media [236]

We wish to reconstruct high-resolution samples of scientific images. Shown below are two microscopic images of physical samples:



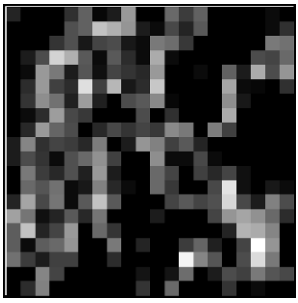
(Microscopic Data from M. Ioannidis, Dept. Chemical Engineering, University of Waterloo)

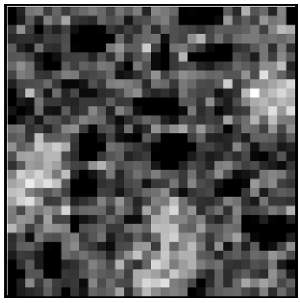
Producing such samples is difficult and expensive, since a physical sample needs to be cut open and carefully polished. Even more troubling is that cutting and polishing may somehow affect the physical sample, leading to distorted images. It would be far preferable to image a physical sample directly in 3D, for example using a MRI. In such a case there is no distortion or cutting of the sample, however the measurements are at a far lower resolution:



Magnetic resonance imagers can be configured in a great variety of ways and it is possible, for example, to measure the diffusivity of water, meaning how tightly the

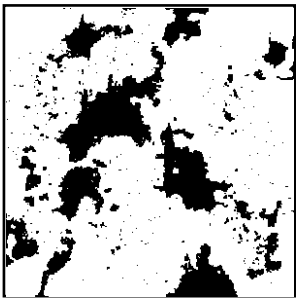
water molecule is constrained, whether in a small pore (tightly bound, dark) or in a much larger one (weakly bound, light):

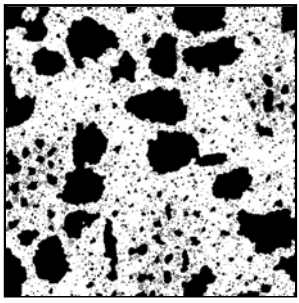




This is essentially a measure of surface-to-volume ratio (or perimeter-to-area in 2D).

The scientists studying such porous materials want a fine-scale image, consistent with the given measurements. We therefore wish to do posterior sampling, given a prior model generated from a high-resolution field, coupled with low-resolution measurements from one or more instruments. The following posterior samples were sampled, using a Gibbs sampler with annealing [236]:





The resulting images are visually a significant improvement over the low-resolution measurements.

Clearly the low-resolution measurements in no way allow us to actually perfectly reconstruct the high-resolution images, however the use of a posterior sampler strikes a fine balance:

- Force the resulting image to be consistent with the measurements, where actually constrained by the measurements;
- Where not constrained by the measurements, generate random samples consistent with the prior model.

That is, much of the very fine-scale detail cannot be inferred from the measurements, but is statistically correct, in the sense that it is consistent with the prior model.

For Further Study

A great deal has been written on Markov Chain Monte Carlo methods, as applied to statistical problems in image processing. The books by Li [207] and by Winkler [335] are both recommended.

Sample Problems

Problem 11.1: Vectorization

Suppose we have a higher-order model, such as thin-plate, on a regular 2D grid. Sketch the pattern of non-interacting state sets, along the lines of Figure 11.8.

Problem 11.2: MRF Sampling

Suppose we are given the "Tree-Bark" kernel from Example 6.1 on page 190, from which we want to generate prior samples. There are four, relatively straightforward approaches for computing a prior sample:

1. By taking the eigendecomposition of P or P^{-1},
2. Using a Cholesky decomposition, as in Section 11.2,
3. Using the FFT, as in Section 11.2.1,
4. Using a marching method and sampling dynamically, as in Section 11.2.2.

Compute $N \times N$ prior samples using each of the above four methods, and discuss the computational complexity as a function of $4 \leq N \leq 256$. Report results only for practical values of N; the eigendecomposition, in particular, will quite rapidly become infeasible.

Problem 11.3: MCMC Sampling

The Ising model is one of the simplest discrete-state models:

1. Implement a basic Gibbs sampler.
2. Implement a vectorized Gibbs sampler, based on a checkerboard approach.

For both methods we initialize with a random binary field, and then iterate the Gibbs sampler. With these basic tools in place, do the following:

(a) Quantify the speed difference, per complete iteration (one sampling of every pixel in the lattice), between the basic and vectorized approaches. Whether there is a difference, and the magnitude of the difference, will likely be highly dependent upon the environment (C, MATLAB, OCTAVE, etc.).
(b) Run the Gibbs sampler for a few different values of coupling $0 \leq \beta \leq 1$.
(c) For highly coupled fields ($\beta \approx 1$), the correct sample is obvious: a constant field (either all +1 or all -1). Not only is this obvious to us, but the energy function H also strongly favours the constant field. How rapidly does the Gibbs sampler approach all constant values?
(d) Explain in some detail why the convergence is so slow in (c), despite the fact that the energy function strongly desires a constant field.
(e) Suggest approaches for accelerating the convergence in (c).

Problem 11.4: Open-Ended Real-Data Problem — Annealing

Typed text on a page does not obey any simple statistical model, since letters have rather complex shapes. However, one prior we *do* have is that each image pixel is binary: black or white.

Let Z be a binary bitmap of text (either captured from the screen, or scanned from a page), and let M_i be the ith measured image, downsampled from Z. The problem is parametrized by

- The degree of downsampling from Z to M_i,
- The number q of measured images $M_i,\ 1 \leq i \leq q$,
- The variance σ^2 of the added measurement noise.

Use Simulated Annealing to solve for estimates $\hat{Z}$. Your solution will need to consider the following:

- The annealing schedule T_k,
- The choice of prior model (for example from Section 7.4),
- The relative weight, in the Gibbs energy, between prior and measurement.

Begin with a modest degree of downsampling (2×2) and a simple prior (Ising).

Part IV

Appendices

A

Algebra

This appendix contains a brief summary of matrix notation, definitions, identities, and common transformations. The following notation will be used throughout:

A	matrix
a_{ij}	the i,jth element of matrix A
$\|A\|$	matrix determinant
A^T	matrix transpose
A^H	matrix conjugate transpose
A^{-1}	matrix inverse
A^+	matrix pseudoinverse
$\mathrm{tr}(A)$	matrix trace
$A_{:}\ [A]_{n\times 1}$	reordering of matrix to column vector
$\|\|A\|\|$	matrix norm

A more comprehensive summary of notation may be found in the Nomenclature section on page XVII.

A.1 Linear Algebra

Our focus in this appendix is on the properties and manipulation of matrices [73, 133, 141, 162, 233, 313]. We can interpret a matrix as a linear operator, transforming from one space to another:

$$\underline{y} = C\underline{x} \quad C \in \mathbb{R}^{k\times n} \quad \Rightarrow \quad C : \underline{x} \in \mathbb{R}^n \longrightarrow \underline{y} \in \mathbb{R}^k, \tag{A.1}$$

thus an understanding of matrices requires an understanding of spaces.

A LINEAR INDEPENDENT set of vectors $\{\underline{x}_i\}, \underline{x}_i \neq \underline{0}$ is one in which none of the vectors can be written as a linear function of the others. If it is possible to express

P. Fieguth, *Statistical Image Processing and Multidimensional Modeling*, Information Science and Statistics, DOI 10.1007/978-1-4419-7294-1_12, © Springer Science+Business Media, LLC 2011

such a relationship, such as

$$\underline{x}_j = \sum_{i \neq j} \alpha_i \underline{x}_i, \tag{A.2}$$

then the set is said to be linearly *dependent*. If the set is linearly *independent*, then from (A.2) it follows that

$$\sum_i \beta_i \underline{x}_i = \underline{0} \quad \Rightarrow \quad \beta_i = 0 \; \forall i. \tag{A.3}$$

A SUBSPACE is a subset V of a vector space such that two properties hold:

1. Origin: $\underline{0} \in V$,
2. Convexity: $\forall \underline{x}_1, \underline{x}_2 \in V, \quad \alpha \underline{x}_1 + \beta \underline{x}_2 \in V \quad \forall \alpha, \beta \in \mathbb{R}$.

We will concern ourselves only with subspaces of multidimensional real vector spaces $\mathbb{R}^n$.

A SPAN is the set of weighted sums induced by a set of vectors:

$$\text{Span}\big(\{\underline{x}_i\}\big) = \sum_i \alpha_i \underline{x}_i \tag{A.4}$$

Any span is a subspace.

A BASIS for a subspace is a linearly-independent set of vectors which span the subspace. The simplest basis elements for the multidimensional real spaces $\mathbb{R}^n$ are the unit vectors

$$\underline{e}_1 = [1\ 0\ 0\ \ldots\ 0\ 0] \quad \cdots \quad \underline{e}_n = [0\ 0\ 0\ \ldots\ 0\ 1]. \tag{A.5}$$

THE DIMENSION $\dim(V)$ of a subspace V is the number of vectors in a basis for V. Thus $\dim\big(\mathbb{R}^n\big) = n$.

THE NULL SPACE of a matrix C is the set of elements which C maps to zero:

$$\text{Nu}(C) = \{\underline{x} | C\underline{x} = \underline{0}\}. \tag{A.6}$$

Every nullspace is a subspace, since

$$\underline{x}_1, \underline{x}_2 \in \text{Nu}(C) \;\longrightarrow\; C(\alpha \underline{x}_1 + \beta \underline{x}_2) = \alpha C \underline{x}_1 + \beta C \underline{x}_2 = \underline{0}. \tag{A.7}$$

The null space is a measure of the degeneracy of C, in the sense of the size of the subspace which is mapped to a single value:

$$C\bar{\underline{x}} = \underline{1} \quad \Rightarrow \quad C\left(\bar{\underline{x}} + \alpha \underline{x}\right) = \underline{1} \;\; \forall \; \underline{x} \in \text{Nu}(C). \tag{A.8}$$

THE RANGE or column-space of a matrix C is the subspace spanned by its column vectors. That is, given

$$C = \left[\underline{c}_1 \ \cdots \ \underline{c}_n\right], \tag{A.9}$$

then $\text{Ra}(C)$, the range of C, is

$$\text{Ra}(C) = \text{Span}(\{\underline{c}_i\}) = \{C\underline{x} \,|\, \underline{x} \in \mathbb{R}^n\}. \tag{A.10}$$

THE RANK of a matrix C equals the number of linearly independent columns of C, or equivalently the number of linearly independent rows. For a matrix $C \in \mathbb{R}^{k\times n}$,

- C has *full rank* if $\text{Rank}(C) = \min(k, n)$,
- If $k < n$, we say that C has *full row rank* if $\text{Rank}(C) = k$,
- If $k > n$, we say that C has *full column rank* if $\text{Rank}(C) = n$.

Because the rank counts the number of linearly independent columns, it is a measure of the dimension of the space into which C projects:

$$\text{Rank}(C) = \dim\big(\text{Ra}(C)\big). \tag{A.11}$$

Similarly, the rank also counts the number of linearly independent rows. Furthermore $\underline{x} \in \text{Nu}(C)$ only if $\underline{x}$ is orthogonal to *all* of the rows of C, leading to the rank-nullity theorem

$$\text{Rank}(C) + \dim\big(\text{Nu}(C)\big) = n. \tag{A.12}$$

Therefore if $\text{Nu}(C) = \underline{0}$ then C must have full column rank.

Rank can never be increased by matrix multiplication, therefore

$$\text{Rank}(AB) \leq \min\big\{\text{Rank}(A), \text{Rank}(B)\big\}. \tag{A.13}$$

Finally, if $C \in \mathbb{R}^{k\times n}, k > n$, has full column rank, then

- $\text{Rank}(C^TC) = n$ and C^TC is a smaller $n \times n$ square, invertible matrix;
- $\text{Rank}(CC^T) = n$ and CC^T is a larger $k \times k$ square, singular matrix.

A.2 Matrix Operations

Matrices may be manipulated in many of the same ways as real scalars, but with certain exceptions. The transpose of a matrix is the reflection of the matrix across its diagonal:

$$B = A^T \quad \Rightarrow \quad b_{i,j} = a_{j,i}. \tag{A.14}$$

Thus the transpose of a column vector is a row vector. The sum of two matrices A, B is an element-by-element sum:

$$C = A + B \quad \Rightarrow \quad c_{i,j} = a_{i,j} + b_{i,j} \tag{A.15}$$

where A and B must be of the same size. The product of two matrices is more complicated,

$$C = A \cdot B \quad \Rightarrow \quad c_{i,k} = \sum_j a_{i,j} b_{j,k} \tag{A.16}$$

where there is an implied condition on the matrix sizes that the number of columns in A must equal the number of rows in B. In general, matrix multiplication does not commute, meaning that $A \cdot B \neq B \cdot A$. We will also have use for element-by-element operations

$$\begin{array}{l} C = A \odot B \\ D = A \oslash B \end{array} \quad \Rightarrow \quad \begin{array}{l} c_{ij} = a_{ij} \cdot bij \\ d_{ij} = a_{ij}/bij \end{array} \tag{A.17}$$

The inverse of square matrix A can be defined, such that

$$B = A^{-1} \quad \Rightarrow \quad A \cdot B = B \cdot A = I = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & 1 \end{bmatrix} \tag{A.18}$$

All non-square and many square matrices do not have an inverse, although more general notions of matrix inverses can be defined, as discussed in Section A.9.

For an $n \times n$ square matrix A, the condition of invertibility is the matrix determinant

$$\det(A) = |A| = \prod_i \lambda_i, \tag{A.19}$$

where the λ_i are the eigenvalues of matrix A, as discussed in Section A.7.1. Then

$$\begin{array}{lll} A \text{ singular} & \Leftrightarrow A^{-1} \text{ does not exist} & A \text{ nonsingular} \Leftrightarrow A^{-1} \text{ exists} \quad \text{(A.20)} \\ & \Leftrightarrow |A| = 0 & \Leftrightarrow |A| \neq 0 \quad \text{(A.21)} \\ & \Leftrightarrow \exists i \ni \lambda_i = 0 & \Leftrightarrow \lambda_i \neq 0 \; \forall i \quad \text{(A.22)} \\ & \Leftrightarrow \text{Nu}(A) \neq \{\underline{0}\} & \Leftrightarrow \text{Nu}(A) = \{\underline{0}\} \quad \text{(A.23)} \\ & \Leftrightarrow \text{Rank}(A) < n & \Leftrightarrow \text{Rank}(A) = n \quad \text{(A.24)} \end{array}$$

Various algorithms are known [141, 162] for computing matrix determinants. It is important to realize, however, that the numerical determination of matrix singularity is very difficult because an infinitesimal perturbation of a singular matrix will make it nonsingular. That is, numerical rounding errors make the *precise* computation of

a determinant difficult, and it is therefore similarly difficult to determine singularity based on a comparison of the computed determinant with zero.

Related to the determinant is the matrix trace. For an $n \times n$ square matrix A, the trace

$$\operatorname{tr}(A) = \sum_{i=1}^{n} a_{ii} = \sum_{i} \lambda_i. \tag{A.25}$$

Because trace and expectation (Appendix B.1.1) are both linear they commute, allowing a trace to simplify certain expectation expressions.

Matrix Addition / Multiplication Identities:

$$A + B = B + A \tag{A.26}$$

$$(A + B)^T = A^T + B^T \qquad (AB)^T = B^T A^T \tag{A.27}$$

Matrix Trace Identities:

$$\operatorname{tr}(A + B) = \operatorname{tr}(A) + \operatorname{tr}(B) \qquad \operatorname{tr}(kA) = k \operatorname{tr}(A) \tag{A.28}$$

$$\operatorname{tr}(AB) = \operatorname{tr}(BA) \qquad \operatorname{tr}(ABC) = \operatorname{tr}(BCA) = \operatorname{tr}(CAB) \tag{A.29}$$

Matrix Determinant Identities:

$$|AB| = |A| \cdot |B| \qquad |A^{-1}| = \frac{1}{|A|} \qquad |kA| = k^n |A| \tag{A.30}$$

$$\det(I - AB) = \det(I - BA) \tag{A.31}$$

Block-Matrix Determinants:

$$\det \begin{bmatrix} a & b \\ c & d \end{bmatrix} = ad - bc \qquad \det \begin{bmatrix} A & B \\ 0 & D \end{bmatrix} = |A| \cdot |D| \tag{A.32}$$

$$\det \begin{bmatrix} A & B \\ C & D \end{bmatrix} = |D| \cdot |A - BD^{-1}C| = |A| \cdot |D - CA^{-1}B| \tag{A.33}$$

Matrix Inversion Identities:

$$(AB)^{-1} = B^{-1}A^{-1} \qquad (A^{-1})^T = (A^T)^{-1} \tag{A.34}$$

$$(I - AB)^{-1}A = A(I - BA)^{-1} \tag{A.35}$$

Block-Matrix Inversions:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{ad-bc}\begin{bmatrix} d & -b \\ -c & a \end{bmatrix} \tag{A.36}$$

$$\begin{bmatrix} A & B \\ \mathbf{0} & D \end{bmatrix}^{-1} = \begin{bmatrix} A^{-1} & -A^{-1}BD^{-1} \\ \mathbf{0} & D^{-1} \end{bmatrix} \tag{A.37}$$

$$\begin{aligned} \begin{bmatrix} A & B \\ C & D \end{bmatrix}^{-1} &= \begin{bmatrix} A^{-1}+A^{-1}BS_A^{-1}CA^{-1} & -A^{-1}BS_A^{-1} \\ -S_A^{-1}CA^{-1} & S_A^{-1} \end{bmatrix} \\ &= \begin{bmatrix} S_D^{-1} & -S_D^{-1}BD^{-1} \\ -D^{-1}CS_D^{-1} & D^{-1}+D^{-1}CS_D^{-1}BD^{-1} \end{bmatrix} \end{aligned} \tag{A.38}$$

where $S_A = (D - CA^{-1}B), S_D = (A - BD^{-1}C)$. Comparing the last two equivalent forms for block-matrix inversion leads to the ABCD lemma

$$(A + BCD)^{-1} = A^{-1} - A^{-1}B(C^{-1} + DA^{-1}B)^{-1}DA^{-1}. \tag{A.39}$$

A.3 Matrix Positivity

For scalar values, notions of positivity, negativity, and relative size are unambiguous. For example,

$$-1 < 0 \qquad 3.5 > 0 \qquad 2.14 < 4.8. \tag{A.40}$$

However, for matrices notions of inequality and positivity are much less clear; matrix element-by-element comparisons are, in most cases, not very useful. Instead, we say that a matrix A is

Positive definite	$A > \mathbf{0}$	If $\underline{x}^T A\underline{x} > 0 \;\; \forall \underline{x} \neq \underline{0}$
Positive semidefinite	$A \geq \mathbf{0}$	If $\underline{x}^T A\underline{x} \geq 0 \;\; \forall \underline{x}$
Negative semidefinite	$A \leq \mathbf{0}$	If $\underline{x}^T A\underline{x} \leq 0 \;\; \forall \underline{x}$
Negative definite	$A < \mathbf{0}$	If $\underline{x}^T A\underline{x} < 0 \;\; \forall \underline{x} \neq \underline{0}$

Given the eigendecomposition (Appendix A.7.1) for A

$$A\underline{v}_i = \underline{v}_i\lambda_i \qquad \text{or} \qquad AV = V\Lambda \tag{A.41}$$

then

$$\underline{x}^T A\,\underline{x} = \underline{x}^T V\Lambda\, V^T\underline{x} = (V^T\underline{x})^T\Lambda\,(V^T\underline{x}) = \underline{z}^T\Lambda\,\underline{z} = \sum_i \lambda_i z_i^2. \tag{A.42}$$

Therefore the positivity of A is directly related to the signs of the eigenvalues of A:

$$A > \mathbf{0} \;\text{ iff }\; \lambda_i > 0\;\forall i \qquad A \geq \mathbf{0} \;\text{ iff }\; \lambda_i \geq 0\;\forall i. \tag{A.43}$$

Matrix inequalities can then be interpreted as the positive definiteness of a matrix difference:

$$A > B \Rightarrow (A - B) > \mathbf{0} \qquad C \leq D \Rightarrow (C - D) \leq \mathbf{0}. \tag{A.44}$$

Positive definiteness identities:

$$A > \mathbf{0} \Rightarrow -A < \mathbf{0} \qquad A \geq \mathbf{0} \Rightarrow -A \leq \mathbf{0} \tag{A.45}$$

$$A > \mathbf{0}, B > \mathbf{0} \Rightarrow A + B > \mathbf{0} \qquad A \geq \mathbf{0}, B \geq \mathbf{0} \Rightarrow A + B \geq \mathbf{0} \tag{A.46}$$

$$A > \mathbf{0} \Rightarrow A^{-1} > \mathbf{0} \qquad A < \mathbf{0} \Rightarrow A^{-1} < \mathbf{0} \tag{A.47}$$

$$A \geq \mathbf{0} \Rightarrow B^T A\, B \geq \mathbf{0} \ \ \forall B \qquad A > \mathbf{0} \Rightarrow B^T A\, B > \mathbf{0} \ \ \forall \text{ invertible } B \tag{A.48}$$

Computing all of the eigenvalues of A is one (computationally demanding) way to test its positive-definiteness. One other method is Sylvester's test [162, 233]:

Given $n \times n$ matrix A, define

$$d_i = \det(A_i) \qquad \text{where} \qquad A_i = \begin{bmatrix} a_{11} & \cdots & a_{1i} \\ \vdots & & \vdots \\ a_{i1} & \cdots & a_{ii} \end{bmatrix}. \tag{A.49}$$

Then A is positive-definite if and only if $d_i > 0$ for all $i = 1, \ldots, n$.

A.4 Matrix Positivity of Covariances

All covariance matrices (Section B.4) must satisfy $\Lambda \geq \mathbf{0}$. The reason for this constraint, and a parallel interpretation of the meaning of matrix positivity and inequalities, is most easily seen in the context of random vectors (Appendix B.1.3). First, given a random vector $\underline{z}$ with covariance Λ, we can calculate the variance of any scalar linear function of $\underline{z}$:

$$\text{var}(\underline{w}^T \underline{z}) = \underline{w}^T \Lambda\, \underline{w}. \tag{A.50}$$

A variance must, by definition, be non-negative, therefore we have

$$\underline{w}^T \Lambda\, \underline{w} \geq 0 \ \ \forall \underline{w} \quad \Rightarrow \quad \Lambda \geq \mathbf{0}. \tag{A.51}$$

That is, all covariances must, by definition, be positive-semidefinite.

Next, let us consider a matrix inequality, in which one matrix is "larger" than the other. Suppose we have two random vectors with corresponding covariances:

$$\underline{x} \sim \Lambda_x \quad \underline{y} \sim \Lambda_y \quad \text{where} \quad \Lambda_x > \Lambda_y. \tag{A.52}$$

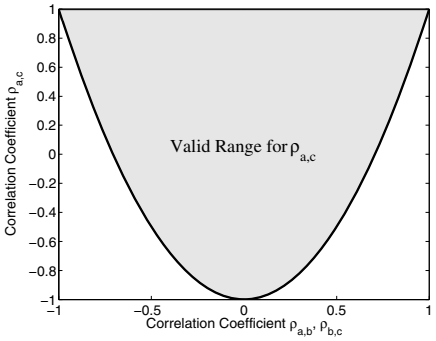


Fig. A.1. The shaded area shows the range of valid correlation coefficients between random variables a, c, given the strength of the relationship with intermediate variable b. If a, b and b, c are uncorrelated ($\rho_{ab} = \rho_{bc} = 0$), then the relationship between a, c is completely unconstrained. As a, b and b, c become more strongly correlated ($|\rho_{ab}| = |\rho_{bc}| \to 1$), the range of valid correlation between a, c becomes increasingly constrained.

The matrix inequality in the covariances implies that

$$\underline{w}^T \Lambda_x \, \underline{w} > \underline{w}^T \Lambda_y \, \underline{w} \qquad \Rightarrow \qquad \mathrm{var}(\underline{w}^T \underline{x}) > \mathrm{var}(\underline{w}^T \underline{y}). \tag{A.53}$$

That is, a "larger" covariance implies a larger variance in *any* linear function of the corresponding random vector.

Finally, the positive-definiteness requirement for a covariance can also be interpreted intuitively. Suppose I have three random variables a, b, c, where the ab and bc correlations are known; what does this imply about the ac correlation?

$$a \overset{\text{correl.}}{\longleftrightarrow} b \quad b \overset{\text{correl.}}{\longleftrightarrow} c \qquad \Longrightarrow \qquad a \overset{???}{\longleftrightarrow} c \tag{A.54}$$

We can ask how $\rho_{a,c}$, the correlation coefficient between a and c, is constrained by the given correlation coefficients $\rho_{a,b}, \rho_{b,c}$. If a, b, c have unit variance, then

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} \sim \begin{array}{|c|c|c|} \hline 1 & \rho_{a,b} & ? \\ \hline \rho_{a,b} & 1 & \rho_{b,c} \\ \hline ? & \rho_{b,c} & 1 \\ \hline \end{array} \tag{A.55}$$

As the $a \Leftrightarrow b$ and $b \Leftrightarrow c$ relationships become stronger (i.e., as $|\rho| \to 1$), the relationship between a and c becomes increasingly constrained. Figure A.1 plots the permitted range of values for $\rho_{a,c}$ as a function of $\rho_{a,b} = \rho_{b,c}$. The range of valid values for $\rho_{a,c}$ are precisely those which preserve the positive-definiteness of the 3×3 covariance in (A.55), and the filling in of such unknown values is called the *covariance extension problem* [48].

A.5 Matrix Types

There exists a wide variety of terms associated with matrices and their properties [141, 162, 163]; the most commonly used are summarized in Table A.1. Except where stated otherwise, the terms all relate to *square* matrices of size $n \times n$. Those definitions which admit a particularly simple visual interpretation are also sketched in Figure A.2.

A.6 Matrix / Vector Derivatives

It is possible to define derivatives of a scalar function $f()$, vector function $\underline{f}()$, or matrix function $F()$ with respect to a scalar x, vector $\underline{x}$, or matrix X. In particular, there are five cases of interest:

Vector derivative of a scalar function or vice-versa:

$$\text{I.}\quad \frac{\partial f}{\partial \underline{x}} = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix} \qquad \text{II.}\quad \frac{\partial \underline{f}}{\partial x} = \begin{bmatrix} \frac{\partial f_1}{\partial x} \\ \vdots \\ \frac{\partial f_k}{\partial x} \end{bmatrix}$$

Derivatives of or by a matrix:

$$\text{III.}\ \frac{\partial f}{\partial X} = \begin{bmatrix} \frac{\partial f}{\partial x_{11}} & \cdots & \frac{\partial f}{\partial x_{1n}} \\ \vdots & & \vdots \\ \frac{\partial f}{\partial x_{k1}} & \cdots & \frac{\partial f}{\partial x_{kn}} \end{bmatrix} \qquad \text{IV.}\ \frac{\partial F}{\partial x} = \begin{bmatrix} \frac{\partial f_{11}}{\partial x} & \cdots & \frac{\partial f_{1n}}{\partial x} \\ \vdots & & \vdots \\ \frac{\partial f_{k1}}{\partial x} & \cdots & \frac{\partial f_{kn}}{\partial x} \end{bmatrix} \tag{A.56}$$

Vector Derivative of a vector function:

$$\text{V.}\quad \frac{\partial \underline{f}}{\partial \underline{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_k}{\partial x_1} & \cdots & \frac{\partial f_k}{\partial x_n} \end{bmatrix}$$

where we have assumed vectors $\underline{f}, \underline{x}$ to be column vectors.

All of these definitions are straightforward, with the exception of case V where it is ambiguous whether the elements of $\underline{f}$ index the columns or the rows. In particular, the transpose operation commutes with differentiation

$$\frac{\partial f}{\partial \underline{x}^T} = \left(\frac{\partial f}{\partial \underline{x}}\right)^T \quad \frac{\partial \underline{f}^T}{\partial x} = \left(\frac{\partial \underline{f}}{\partial x}\right)^T \quad \frac{\partial f}{\partial X^T} = \left(\frac{\partial f}{\partial X}\right)^T \quad \frac{\partial F^T}{\partial x} = \left(\frac{\partial F}{\partial x}\right)^T \tag{A.57}$$

in all cases *except* case V. As a further complication, there is no universal agreement on row–column conventions for cases I and V, and unfortunately different authors

Term	Definition / Comments
* Diagonal	$a_{ij} = 0, i \neq j$
* Upper Triangular	$a_{ij} = 0, i < j$ (Strictly Upper for $i \leq j$)
* Lower Triangular	$a_{ij} = 0, i > j$ (Strictly Lower for $i \geq j$)
* Symmetric	$a_{ij} = a_{ji}$ or $A = A^T$
* Skew-Symmetric	$a_{ij} = -a_{ji}$ or $A = -A^T$
* Hermitian	$a_{ij} = a^*_{ji}$ or $A = A^H$
Singular	$\lvert A \rvert = 0$
Special	$\lvert A \rvert = 1$
Normal	$AA^H = A^H A$
Orthogonal	$AA^T = A^T A = I$
Unitary	$AA^H = A^H A = I$
* Circulant	$a_{ij} = a_{(i+k) \bmod n, (j+k) \bmod n} \quad \forall k$
Diagonalizable	BAB^{-1} is diagonal for some B
Similar	A similar to C if $C = BAB^{-1}$ for some B
Unitarily Similar	A unitarily similar to C if $C = BAB^{-1}$, unitary B
Positive-Definite	$\underline{x}^H A \underline{x} > 0$ for all $\underline{x} \neq \underline{0}$
Positive-Semidefinite	$\underline{x}^H A \underline{x} \geq 0$ for all $\underline{x}$
Idempotent	$AA = A$
Nilpotent	$A^k = 0$ for some positive integer k
Involutary	$AA = I$
* Stochastic	$\sum_j a_{ij} = 1$
* Selection	Each row has one 1.0, the rest zeros
* Permutation	Each row and column has one 1.0, the rest zeros
* Toeplitz	A matrix with constant diagonals
* Hankel	A matrix with constant anti-diagonals
* Hadamard	A binary matrix $a_{ij} = \pm 1$, where $AA^T = A^T A = nI$
* Hilbert	$a_{ij} = 1/(i + j - 1)$
* Pascal	An integer matrix with integer inverse
* Vandermonde	Each row is a geometric sequence: $a_{ij} = \alpha_i^{n-j}$
Metzler	$a_{i,j} \geq 0$ for $i \neq j$
Householder	Any $n \times n$ matrix $(I - 2\underline{v}\,\underline{v}^T / (\underline{v}^T \underline{v}))$, $\underline{v}$ nonzero
* Jordan	A block diagonal matrix with Jordan blocks
Jordan Block	$a_{i,j} = 0$ except $a_{i,i} = \alpha,\ a_{i,i+1} = 1$
* Wronskian	The zeroth to $(n-1)$ derivatives of a vector function
* Jacobian	The vector derivative of a vector function
* Hessian	The second vector derivative of a scalar function

Table A.1. Common named matrix types (starred entries sketched in Figure A.2)

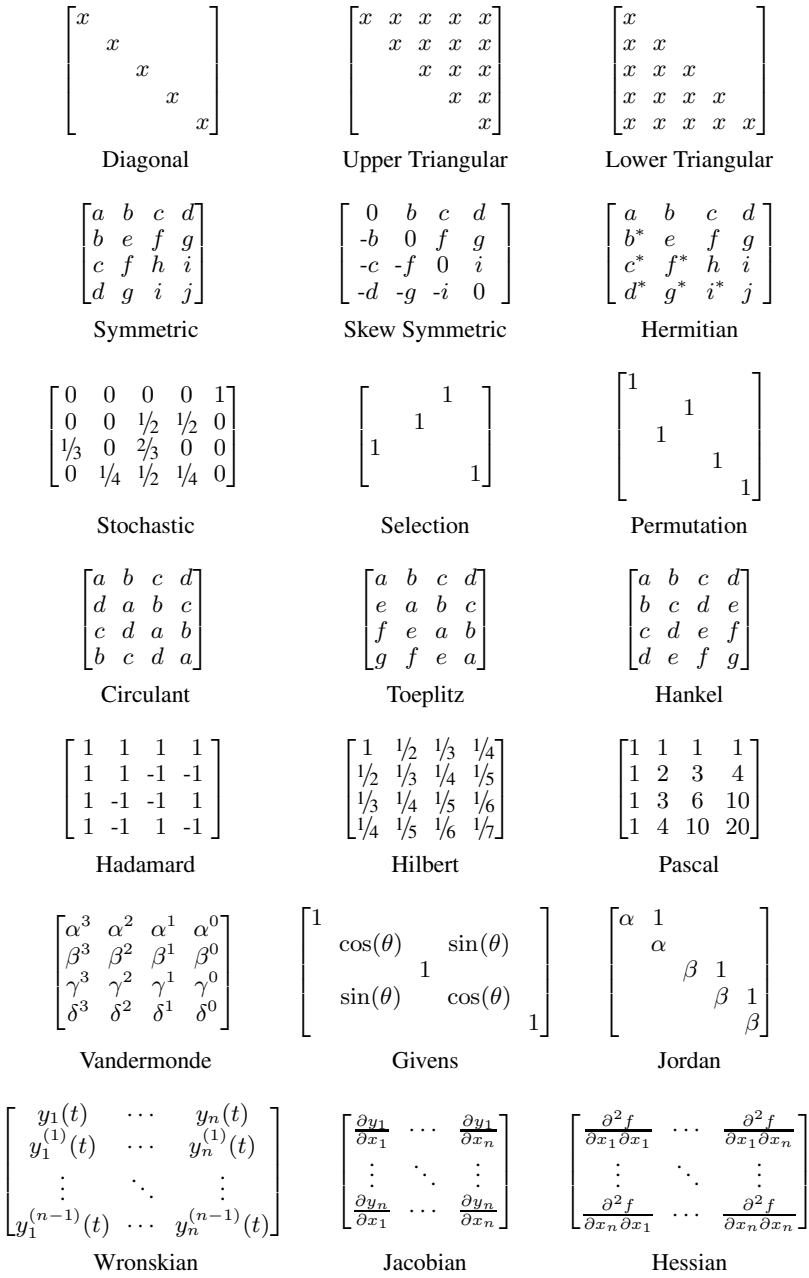


Fig. A.2. Examples of commonly used matrix types from Table A.1. A blank area in a matrix implies zeros. The figure lists only illustrative examples.

Case	Derivatives of Linear Functions
V	$\frac{\partial}{\partial \underline{x}} \underline{x} = I$
I	$\frac{\partial}{\partial \underline{x}} (\underline{a}^T \underline{x}) = \underline{a}$
V	$\frac{\partial}{\partial \underline{x}} A\underline{x} = A$
V	$\frac{\partial}{\partial \underline{x}} A\underline{y} = A \frac{\partial}{\partial \underline{x}} \underline{y}$
III	$\frac{\partial}{\partial X} (\underline{a}^T X \underline{b}) = \underline{a}\, \underline{b}^T$
IV	$\frac{\partial}{\partial \underline{x}} (YZ) = Y \frac{\partial Z}{\partial \underline{x}} + \frac{\partial Y}{\partial \underline{x}} Z$

Case	Derivatives of Quadratic Functions
I	$\frac{\partial}{\partial \underline{x}} (\underline{x}^T \underline{x}) = 2\underline{x}^T$
I	$\frac{\partial}{\partial \underline{x}} (\underline{x}^T A\, \underline{x}) = \underline{x}^T (A + A^T)$
I	$\frac{\partial}{\partial \underline{x}} (A\underline{x} + \underline{b})^T Q (A\underline{x} + \underline{b}) = 2A^T (A\underline{x} + \underline{b})$
III	$\frac{\partial}{\partial X} (\underline{a}^T X^T X \underline{b}) = X(\underline{a}\, \underline{b}^T + \underline{b}\, \underline{a}^T)$
III	$\frac{\partial}{\partial X} (\underline{a}^T X^T X \underline{a}) = 2X \underline{a}\, \underline{a}^T$
III	$\frac{\partial}{\partial X} (\underline{a}^T X^T C X \underline{b}) = C^T X \underline{a}\, \underline{b}^T + C X \underline{b}\, \underline{a}^T$

Table A.2. Vector and matrix derivatives of linear and quadratic functions [46, 257]

may use opposing conventions, thus the derivatives presented here may, in some cases, need to be transposed to fit with other conventions. A summary of useful derivatives is shown in Tables A.2 and A.3.

There are three matrices commonly connected with derivatives: the Jacobian, the Wronskian, and the Hessian, all of which are illustrated in Figure A.2:

Jacobian: If $\underline{y}$ is a vector function of $\underline{x}$, then the derivative $d\underline{y}^T / d\underline{x}$ is the Jacobian matrix of $\underline{y}$ with respect to $\underline{x}$. The determinant $|d\underline{y}^T / d\underline{x}|$ is also referred to as the Jacobian, essentially a normalizing constant in the change of variables from $\underline{x}$ to $\underline{y}$.

Case	Derivatives of Matrix Inverses
IV	$\frac{\partial}{\partial \underline{x}} Y^{-1} = -Y^{-1} \frac{\partial Y}{\partial \underline{x}} Y^{-1}$
III	$\frac{\partial}{\partial X} (\underline{a}^T X^{-1} \underline{b}) = -X^{-T} \underline{a}\, \underline{b}^T X^{-T}$

Case	Derivatives of Matrix Determinants
III	$\frac{\partial}{\partial X} \lvert X \rvert = \lvert X \rvert \cdot X^{-T}$
III	$\frac{\partial}{\partial X} \lvert AXB \rvert = \lvert AXB \rvert \cdot X^{-T}$
III	$\frac{\partial}{\partial X} \ln(\lvert AXB \rvert) = A^T (AXB)^{-T} B^T = X^{-T}$

Case	Derivatives of Matrix Traces
III	$\frac{\partial}{\partial X} \text{tr}(X) = \frac{\partial}{\partial X} \text{tr}(X^T) = I$
III	$\frac{\partial}{\partial X} \text{tr}(XA) = \frac{\partial}{\partial X} \text{tr}(AX) = A^T$
III	$\frac{\partial}{\partial X} \text{tr}(AXB) = A^T B^T$
III	$\frac{\partial}{\partial X} \text{tr}(XAX^T) = X(A + A^T)$
III	$\frac{\partial}{\partial X} \text{tr}(X^{-1}) = -X^{-T} X^{-T}$

Table A.3. Vector and matrix derivatives of matrix inverses, determinants, and traces [46, 257]

Wronskian: If $\underline{y}$ is a vector function of scalar t, then the matrix built by assembling the rows $\underline{y}^{(0)}(t)\ \ldots\ \underline{y}^{(n-1)}(t)$ is known as a Wronskian of $\underline{y}$.

Hessian: If f is a scalar function of vector $\underline{x}$, then the second-derivative matrix $d/d\underline{x}^H (df/d\underline{x})$ is known as the Hessian matrix of $f(\underline{x})$. The positive-definiteness of the Hessian relates to the extremal properties (minimum, maximum, or saddle point) of the function.

A.7 Matrix Transformations

A wide variety of matrix transformations exist [141], many of which are used in solving linear systems, normal equations, or least-squares problems, all of which

Page	*Transformation*	*Matrix Assumptions*	*Purpose of Transformation*
396	Eigendecomp.	Square	Matrix diagonalization
400	SVD	None	Matrix orthogonalization
401	Cholesky	Positive-Definite	Square roots, Linear systems, Least squares
402	Gauss	Any	Matrix element zeroing
402	Gauss Elimin.	Nonsingular	Linear systems, Matrix inversion
403	LU	Square Matrix representation	Linear systems, Matrix inversion,
404	Gram–Schmidt	Any	Vector orthogonalization
405	Householder	Any	Matrix element zeroing
406	Givens	Any	Matrix element zeroing
406	QR	Full-column rank	Representation, Least squares
407	Schur	Square	Representation

Table A.4. An overview of the matrix transformations discussed in Section A.7

are of interest in this book, particularly in Chapters 8 and 9. The transformations discussed in this section are listed in Table A.4.

A.7.1 Eigendecompositions

The eigendecomposition is one of the most fundamental and powerful matrix tools in all of linear algebra.

For any square matrix A, an eigenvector $\underline{v}$ and corresponding eigenvalue λ must satisfy

$$A\underline{v} = \underline{v}\lambda. \tag{A.58}$$

That is, the eigenvectors point in those special directions which are invariant to the repeated application of linear operator A, greatly simplifying analysis:

$$\underbrace{A \cdot \ldots \cdot A}_{q \text{ times}}\underline{v} = A^q\underline{v} = A^{q-1}(\underline{v}\lambda) = \underline{v}\lambda^q. \tag{A.59}$$

Since the eigenvalues determine many of the properties of a matrix, the distribution of eigenvalues $\{\lambda_i\}$ is known as the *spectrum* of a matrix.

For an $n \times n$ matrix A there are n eigenvalues and eigenvectors, however for certain matrices there can be redundancy (multiplicity) among them. However, if A is real and symmetric (such as a covariance), then the n eigenvalues are real and the n eigenvectors are linearly independent and orthogonal. That is, the eigenvectors form a basis for $\mathbb{R}^n$, allowing *any* n-dimensional vector $\underline{x}$ to be expressed in terms of the eigenvectors:

$$\underline{x} = \sum_{i=1}^{n} \alpha_i \underline{v}_i, \tag{A.60}$$

where the orthogonality of the eigenvectors allows the weights to be easily calculated:

$$\alpha_i = \frac{\underline{v}_i^T \underline{x}}{\underline{v}_i^T \underline{v}_i}. \tag{A.61}$$

The transformation (A.60) simplifies the analysis for linear operations on any vector:

$$\underbrace{A \cdot \ldots \cdot A}_{q \text{ times}} \underline{x} = A^q \sum_{i=1}^{n} \alpha_i \underline{v}_i = \sum_{i=1}^{n} \alpha_i \lambda_i^q \underline{v}_i. \tag{A.62}$$

Normalizing the eigenvectors to unit length, we can write the n eigenvectors and eigenvalues in matrix form

$$V = \begin{bmatrix} \underline{v}_1 & \cdots & \underline{v}_n \end{bmatrix} \qquad \Lambda = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \lambda_n \end{bmatrix} \tag{A.63}$$

such that Λ is a diagonal matrix and V is an orthogonal matrix

$$V^T V = V V^T = I. \tag{A.64}$$

With this notation the eigendecomposition (A.58) can be rewritten as

$$A\underline{v}_i = \underline{v}_i \lambda_i \qquad \Rightarrow \qquad AV = V\Lambda, \tag{A.65}$$

from which it is particularly easy to derive the similarity transformation for matrix diagonalization and related expressions:

$$V^T A V = \Lambda \qquad A = V \Lambda V^T \qquad A^{-1} = V \Lambda^{-1} V^T \qquad A^q = V \Lambda^q V^T. \tag{A.66}$$

Fundamentally, eigendecompositions are about taking a coupled linear system and decoupling it into individual modes (the eigenvectors) which evolve independently, and where the associated eigenvalue describes the behaviour of the associated mode. For example, suppose we have a mechanical system of n masses connected by springs:

$$\ddot{\underline{x}}(t) = K \underline{x}(t). \tag{A.67}$$

This is just a rewriting of Newton's $F = ma$, where the acceleration ($\ddot{\underline{x}}$) is written in terms of force/mass ($K\underline{x}$). By finding the eigendecomposition of K,

$$K = V_K \Lambda_K V_K^T, \tag{A.68}$$

the coupled problem of (A.67) can be transformed

$$\underline{y}(t) = V_K^T \underline{x}(t) \quad \Rightarrow \quad \ddot{y}_i(t) = \lambda_i y_i(t), \tag{A.69}$$

a set of n single-spring/mass systems whose solution is simple:

$$y_i(t) = y_i(0) \cos\left(t\sqrt{-\lambda_i}\right). \tag{A.70}$$

Further illustrations are given below for dynamic and covariance matrices.

The significance and power of the eigendecomposition should make it no surprise that there are many related concepts and extensions. Related concepts include the Cholesky and QR decompositions (Appendix A.7.3), positive definiteness (Appendix A.3), and matrix square roots (Appendix A.8).

Extensions include the Jordan form for non-diagonalizable matrices (not discussed here), the Schur decomposition for complex matrices (Appendix A.7.3), the singular value decomposition for rectangular matrices (Appendix A.7.2), and generalized eigendecompositions, in which we seek solutions for $\underline{v}, \lambda$ to

$$A\underline{v} = B\underline{v}\lambda, \tag{A.71}$$

where normally B is not invertible, a context which can arise in singular estimation problems, but which does not arise in this book.

Eigendecomposition and Dynamic Matrices

If $n \times n$ matrix A is a dynamic matrix, describing the evolution of $\underline{x}$ as

$$\underline{x}(t+1) = A\underline{x}(t), \tag{A.72}$$

then the stability of the iteration is determined by the eigenvalues of A. Specifically, the *spectral radius*, the largest eigenvalue magnitude

$$\rho(A) = \max_i |\lambda_i|, \tag{A.73}$$

satisfies $\rho(A) < 1$ for stable systems and $\rho(A) > 1$ for unstable ones.

Given an initial condition $\underline{x}(0)$, then $\underline{x}(t) = A^t\underline{x}(0)$ is an iterative calculation, with a complexity increasing with t. Instead, the eigendecomposition $A = V_A \Lambda_A V_A^T$ allows the coupled dynamics of (A.72) to be decoupled into individual modes, based on the eigenvectors, which evolve independently, thus

$$\begin{array}{ccc} \underline{x}(0) & \xrightarrow{V_A^T} & \underline{y}(0) = V_A^T \underline{x}(0) \\ & & \downarrow \\ \underline{x}(t) & \xleftarrow{V_A} & \underline{y}(t) = \Lambda_A^t \underline{y}(0) \end{array} \tag{A.74}$$

leads to a closed-form solution $\underline{x}(t) = V\Lambda^t V^T \underline{x}(0)$, where the complexity of computing Λ^t is fixed, since Λ is diagonal.

If the dynamics matrix describes the evolution of an error,

$$\underline{e}(t+1) = Q\underline{e}(t) \tag{A.75}$$

such as in the iterative linear-system solvers of Chapter 9, then we are interested in the rate at which the error decays to zero. Because

$$\underline{e}(t) = Q^t \underline{e}(0) \tag{A.76}$$

from (A.62) we know that

$$\underline{e}(t) = Q^t \underline{e}(0) = \sum_{i=1}^{n} \left(\underline{v}_i^T \underline{e}(0)\right) \lambda_i^t \underline{v}_i. \tag{A.77}$$

That is, each eigenvector $\underline{v}_i$ describes the shape or form of the error which decays at a rate controlled by λ_i.

Eigendecomposition and Covariance Matrices

In the context of this book we are frequently concerned with the interrelationship of variables in a random vector, a set of relationships described by a covariance matrix (Appendix B.1.3). Given a set of n coupled random variables, the eigentransformation decouples them:

$$\underline{x} \sim P \quad \Rightarrow \quad \underline{y} = V_P^T \underline{x} \sim \Lambda_P. \tag{A.78}$$

If the joint distribution $p(\underline{x})$ is Gaussian (Appendix B.3), then the eigenvectors of P point along the principal axes of the ellipsoid characterizing the multivariate distribution, and the eigenvalues represent the variances along those directions.

The spectrum, or distribution of eigenvalues, is useful in at least two ways:

- Even for non-Gaussian distributions, the relative sizes of the eigenvalues of a covariance give an indication of the degree to which a multivariate distribution is constrained in various directions.
- The conditioning of matrix P, which strongly affects numerical stability and rates of iterative convergence, is a function of the largest and smallest eigenvalues. As the smallest eigenvalue approaches zero, ever-smaller numeric perturbations lead to matrix corruption and a failure of positive-definiteness.

A.7.2 Singular Value Decomposition

Given any real matrix A, the Singular Value Decomposition (SVD) expresses A as

$$A = USV^T, \tag{A.79}$$

where U, V are orthogonal matrices and where S is a diagonal matrix, the same dimensions as A, containing the singular values along the diagonal:

$$S = \begin{bmatrix} \sigma_1 & & \mathbf{0} \\ & \sigma_2 & \\ \mathbf{0} & & \ddots \\ \hline & \mathbf{0} & \end{bmatrix} \quad \text{or} \quad S = \left[\begin{array}{ccc|c} \sigma_1 & & \mathbf{0} & \\ & \sigma_2 & & \mathbf{0} \\ \mathbf{0} & & \ddots & \end{array}\right] \tag{A.80}$$

where, by convention, the singular values are ordered $\sigma_1 \geq \sigma_2 \geq \cdots \geq 0$.

For real symmetric matrices the SVD and eigendecomposition are nearly equivalent. Representing a covariance A in both ways,

$$A = \underbrace{USV^T}_{\text{SVD}} \qquad \text{and} \qquad A = \underbrace{Q\Lambda Q^T}_{\text{Eig.}} \tag{A.81}$$

by equating the SVD and eigendecomposition expressions we find that

$$\underline{u}_i = \underline{q}_i \qquad \sigma_i = |\lambda_i| \qquad \underline{v}_i = \text{sign}(\lambda_i)\underline{q}_i. \tag{A.82}$$

That is, requiring the singular values to be positive causes the sign of the eigenvalue to be absorbed into one of the orthogonal matrices U, V.

For real *nonsymmetric* matrices the SVD and eigendecompositions are quite different. Whereas many matrices may not diagonalize or may have complex eigenvalues, *every* matrix has a SVD with real, positive singular values.

The behaviour of the SVD is most easily explained in the relationship between two zero-mean random vectors $\underline{x}, \underline{y}$. First, the eigendecomposition of a covariance matrix

$$\underline{x} \sim E\left[\underline{x}\,\underline{x}^T\right] = V\Lambda V^T \quad \Rightarrow \quad E\left[(V^T\underline{x})(V^T\underline{x})^T\right] = \Lambda \tag{A.83}$$

identifies the principal components (Section 8.2.1) of $\underline{x}$: the set of mutually decorrelated random variables $\underline{y} = V^T\underline{x}$ from $\underline{x}$ such that

$$\begin{aligned} &\underline{v}_i^T\,\underline{x} \text{ is uncorrelated with } \underline{v}_j^T\,\underline{x}\ \forall\ j \neq i \\ &\underline{v}_i^T\,\underline{x} \text{ has a variance of } \lambda_i. \end{aligned} \tag{A.84}$$

In contrast, the singular value decomposition of a *cross*-covariance

$$E\left[\underline{x}\,\underline{y}^T\right] = USV^T \quad \Rightarrow \quad E\left[(U^T\underline{x})(V^T\underline{y})^T\right] = S \tag{A.85}$$

identifies the principal components of the relationship *between* $\underline{x}$ and $\underline{y}$. In particular, given $\underline{x} \in \mathbb{R}^n, \underline{y} \in \mathbb{R}^k, k < n$, it follows that

$$\begin{aligned}
&\underline{u}_i^T\,\underline{x},\ i \le k \text{ is uncorrelated with } \underline{v}_j^T\,\underline{y}\ \forall\ j \neq i \\
&\underline{u}_i^T\,\underline{x},\ i > k \text{ is uncorrelated with } \underline{q}^T\,\underline{y}\ \forall\ \underline{q} \\
&\underline{u}_j^T\,\underline{x},\ i \le k \text{ is correlated with } \quad \underline{v}_j^T\,\underline{y} \text{ with strength or significance } \sigma_i.
\end{aligned} \tag{A.86}$$

For a square matrix A, the SVD can be used to calculate $\kappa(A)$, the *condition number* of A, which measures the closeness of A to singularity:

$$\kappa(A) = \frac{\sigma_{\max}(A)}{\sigma_{\min}(A)} \tag{A.87}$$

Thus for covariance matrices, which are symmetric, the equivalence (A.82) between the SVD and the eigendecomposition allows the condition number also to be evaluated via eigenvalues:

$$\kappa(A) = \frac{\max_i |\lambda_i(A)|}{\min_j |\lambda_j(A)|} \tag{A.88}$$

This latter form may be useful in stationary cases where the FFT can be used to calculate eigenvalues, or for specific priors for which the eigendecomposition is known analytically.

A.7.3 Cholesky, Gauss, LU, Gram–Schmidt, QR, Schur

The following ten matrix transformations are widely used and referred to. They are summarized here only very briefly; for a much more extensive discussion the reader is referred to the comprehensive text by Golub and Van Loan [141].

Cholesky Decomposition

Given an $n \times n$ positive-definite symmetric matrix A,

> *The Cholesky decomposition finds a lower-triangular*[1] *matrix* Γ *with positive diagonal elements, such that*
>
> $$A = \Gamma\Gamma^T. \tag{A.89}$$

[1] Note: the default matrix returned by `chol(A)` in MATLAB is upper-triangular. To obtain the lower-triangular form, as in (A.89), use `chol(A,'lower')`.

Clearly Γ is the matrix square root (Appendix A.8) of A; also the triangularity of Γ implies that Γ is easy to invert, leading to an efficient solution for the matrix inverse A^{-1}.

The most attractive aspect of the Cholesky decomposition is that fast, numerically-stable algorithms exist to compute it (see Algorithm 4 in Chapter 10). As all matrix covariances are symmetric positive-definite, the Cholesky decomposition is widely used in statistical processing (particularly for covariance inversion and square roots, as in Appendix A.8).

Gauss Transformation

The Gauss transformation sets to zero all elements in a vector beyond some index i:

An $n \times n$ lower-triangular matrix G_i is a Gauss transform if

$$G_i \underline{x} = G_i \begin{bmatrix} x_1 \\ \vdots \\ x_i \\ x_{i+1} \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} x_1 \\ \vdots \\ x_i \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{A.90}$$

The effect of G_i is to subtract multiples of row i from all following rows, specifically to subtract the multiple x_k/x_i times row i from row k. The Gauss transformation is the elemental step in Gaussian elimination.

Gaussian Elimination

A direct, non-iterative approach to the solving of linear systems and matrix inversions can be realized by the repeated application of Gauss transformations:

Given a linear system $A\underline{x} = \underline{b}$, by applying Gauss transformations we zero the elements in A, such that in the resulting system $U\underline{x} = \underline{\bar{b}}$ the matrix U is upper triangular.

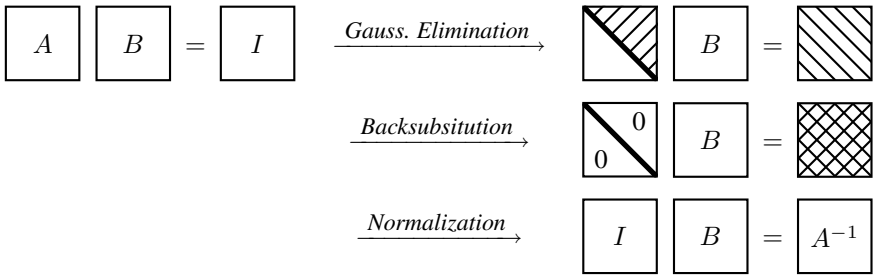
For example, given the linear system, written in equation or matrix form,

$$\begin{aligned} x_1 + 2x_2 &= 3 \\ x_1 + \ \ x_2 &= 5 \end{aligned} \quad \longleftrightarrow \quad \begin{bmatrix} 1 & 2 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 3 \\ 5 \end{bmatrix} \tag{A.91}$$

then by subtracting row one from row two we have

$$\begin{aligned} x_1 + 2x_2 &= 3 \\ -x_2 &= 2 \end{aligned} \quad \longleftrightarrow \quad \begin{bmatrix} 1 & 2 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \end{bmatrix} \tag{A.92}$$

To perform matrix inversion we apply Gaussian elimination to the linear system $AB = I$ where, given A, we seek the solution to $B = A^{-1}$:



(A.93)

The above presents only the basic, conceptual algorithm. There is a wide variety of practical considerations, such as row reordering (pivoting) for numerical stability and specialized algorithms for symmetric matrices.

LU Decomposition

LU factors a matrix into a product of lower and upper-triangular matrices:

The LU decomposition of a matrix A is

$$A = LU, \tag{A.94}$$

where L is lower-triangular and U is upper-triangular.

By definition, after applying Gaussian elimination to A we have an upper triangular product

$$\cdots G_3\Big(G_2\big(G_1 A\big)\Big) = U, \tag{A.95}$$

therefore

$$A = \big(G_1^{-1} G_2^{-1} G_3^{-1} \cdots\big)\, U = LU, \tag{A.96}$$

where the product $G_1^{-1} G_2^{-1} G_3^{-1} \cdots$ is lower-triangular because each of the Gaussian transformations G_i is lower-triangular.

It is important to realize that not every matrix admits an LU decomposition. However, for matrices which admit such a decomposition, the solving of linear systems is extremely efficient:

$$A\underline{x} = \underline{b} \quad \Rightarrow \quad L\big(U\underline{x}\big) = \underline{b} \quad \Rightarrow \quad L\underline{y} = \underline{b},\ U\underline{x} = \underline{y} \tag{A.97}$$

where the triangular form of L, U allows the final equations to be solved easily by backsubstitution.

Gram–Schmidt Orthogonalization

Gram–Schmidt is the orthogonalization of a set of linearly independent vectors:

Given linearly independent $\underline{a}_1, \ldots, \underline{a}_n$*, find vectors* $\underline{b}_1, \ldots, \underline{b}_n$ *such that*

$$\text{Span}(\underline{a}_1, \ldots, \underline{a}_n) = \text{Span}(\underline{b}_1, \ldots, \underline{b}_n) \quad \underline{b}_i^T \underline{b}_j = 0 \; \forall \; i \neq j. \tag{A.98}$$

Alternatively, stated in matrix form, given A *with full column rank, find matrix* B *such that*

$$\text{Ra}(A) = \text{Ra}(B) \quad B^T B \text{ is diagonal.} \tag{A.99}$$

The algorithm proceeds recursively, orthogonalizing a vector by removing components aligned with any previous vector:

$$\begin{aligned} \underline{b}_1 &= \underline{a}_1 \\ \underline{b}_2 &= \underline{a}_2 - \frac{\underline{a}_2^T \underline{b}_1}{\underline{b}_1^T \underline{b}_1} \underline{b}_1 \\ \underline{b}_3 &= \underline{a}_3 - \frac{\underline{a}_3^T \underline{b}_2}{\underline{b}_2^T \underline{b}_2} \underline{b}_2 - \frac{\underline{a}_3^T \underline{b}_1}{\underline{b}_1^T \underline{b}_1} \underline{b}_1 \\ &\vdots \end{aligned} \tag{A.100}$$

Alternative, numerically-robust forms of this method exist [141].

Conjugate Gram-Schmidt Orthogonalization

A modification of the preceding Gram–Schmidt procedure to conjugate-orthogonalize a set of linearly independent vectors:

Given matrix M *and linearly independent* $\underline{a}_1, \ldots, \underline{a}_n$*, find* $\underline{b}_1, \ldots, \underline{b}_n$ *such that*

$$\text{Span}(\underline{a}_1, \ldots, \underline{a}_n) = \text{Span}(\underline{b}_1, \ldots, \underline{b}_n) \quad \underline{b}_i^T A \, \underline{b}_j = 0 \; \forall \; i \neq j. \tag{A.101}$$

The conjugate algorithm proceeds similarly to usual Gram–Schmidt, with vector conjugacy assessed in a reshaped space, modified by M:

$$\underline{b}_1 = \underline{a}_1 \tag{A.102}$$

$$\underline{b}_2 = \underline{a}_2 - \frac{\underline{a}_2^T M \underline{b}_1}{\underline{b}_1^T M \underline{b}_1} \underline{b}_1 \tag{A.103}$$

$$\underline{b}_3 = \underline{a}_3 - \frac{\underline{a}_3^T M \underline{b}_2}{\underline{b}_2^T M \underline{b}_2} \underline{b}_2 - \frac{\underline{a}_3^T M \underline{b}_1}{\underline{b}_1^T M \underline{b}_1} \underline{b}_1 \tag{A.104}$$

$$\vdots$$

Such matrix-conjugate vectors are of key interest in conjugate gradient and related Krylov methods (discussed in Section 9.2.3).

Householder Transformation

The Householder transformation reflects a vector (or matrix column) across a hyperplane to set to zero all but one element of the vector:

An $n \times n$ matrix

$$H = I - 2\frac{\underline{v}\,\underline{v}^T}{\underline{v}^T\underline{v}} \quad \underline{v} \neq \underline{0} \tag{A.105}$$

is called a Householder matrix or reflection, such that $H\underline{x}$ reflects $\underline{x}$ across $\text{Span}(\underline{v})^\perp$*, that is, across the hyperplane with normal vector $\underline{v}$.*

Suppose that we wish to set to zero all except the first element of a vector $\underline{x}$ (normally the column of a matrix). That is, we wish to find a transformation

$$H\underline{x} = \left(I - 2\frac{\underline{v}\,\underline{v}^T}{\underline{v}^T\underline{v}} \right) \underline{x} = \beta \underline{e}_1 \tag{A.106}$$

such that $H\underline{x}$ is a multiple of the first unit vector. This is accomplished by setting

$$\underline{v} = \underline{x} \pm \left(\underline{x}^T \underline{x} \right) \underline{e}_1. \tag{A.107}$$

The appeal of the Householder reflection lies in the simple form of $\underline{v}$.

Note that in applying this transformation to all of the columns of a matrix, the Householder matrix H is never explicitly computed, rather the transformation is calculated directly from $\underline{v}$ [141].

Givens Rotation

A Givens rotation sets a single vector (or matrix column) element to zero:

An $n \times n$ Givens matrix $G(i, j, \theta)$ rotates the (i, j) vector elements by angle θ, where θ is normally selected such that

$$\underline{y} = G(i, j, \theta)\underline{x} \;\Rightarrow\; y_j = 0. \tag{A.108}$$

A Givens matrix $G(i, j, \theta)$ is equal to the identity I, except for a 2×2 rotation matrix in the four elements indexed by i, j:

$$\begin{array}{|cc|l} \cos(\theta) & \sin(\theta) & \text{Row } i \\ -\sin(\theta) & \cos(\theta) & \text{Row } j \end{array} \qquad \begin{array}{c} \\ \\ \end{array} \tag{A.109}$$

$$\text{Column } i \quad \text{Column } j$$

Clearly only elements i, j of a vector are affected in the product

$$\underline{y} = G(i, j, \theta)\underline{x} \;\Rightarrow\; y_k = \begin{cases} x_k & k \neq i, j \\ x_i \cos(\theta) - x_j \sin(\theta) & k = i \\ x_i \sin(\theta) + x_j \cos(\theta) & k = j \end{cases} \tag{A.110}$$

We can set the jth element to zero by selecting

$$\cos(\theta) = \frac{x_i}{\sqrt{x_i^2 + x_j^2}} \qquad \sin(\theta) = -\frac{x_j}{\sqrt{x_i^2 + x_j^2}} \tag{A.111}$$

QR Decomposition

The QR decomposition expresses a matrix as the product of orthogonal and upper-triangular matrices:

The QR factorization of a $k \times n$ matrix A is given by

$$A = QR, \tag{A.112}$$

where Q is orthogonal and R is upper triangular.

The QR decomposition, a relatively complicated algorithm, can be accomplished using the Householder, Givens, or Gram–Schmidt transformations, and plays a central role in computing Schur decompositions.

Schur Decomposition

The Schur decomposition or factorization of a matrix is an alternative to the eigendecomposition:

> *Given a real, square matrix A, the Schur factorization of A is*
>
> $$A = QUQ^T, \tag{A.113}$$
>
> *where Q is orthogonal and U is an upper-triangular matrix (possibly with diagonal blocks) with the eigenvalues of A appearing along the diagonal of U.*
>
> *Given a complex, square matrix A, the Schur factorization of A is*
>
> $$A = ZUZ^H, \tag{A.114}$$
>
> *where Z is unitary and U is as before.*

Clearly if U is a diagonal matrix, then the Schur form is equivalent to the regular eigendecomposition.

A.8 Matrix Square Roots

In general, if a matrix P can be expressed as

$$P = \Gamma^T \Gamma \tag{A.115}$$

then Γ is defined as a matrix square root of P. This expression is possible for all positive-semidefinite P (that is, for *all* covariance matrices), however the choice of Γ is *not* unique. Indeed, any orthogonal transformation of Γ remains a square root:

$$\bar{\Gamma} = U\Gamma \qquad \Rightarrow \qquad \bar{\Gamma}^T\bar{\Gamma} = \Gamma^T U^T U \Gamma = \Gamma^T\Gamma = P \tag{A.116}$$

where U is any orthogonal matrix, $U^T U = I$.

The matrix square root is most easily expressed in terms of its eigendecomposition:

$$P = V\Lambda V^T = \left(V\Lambda^{1/2}\right)\left(V\Lambda^{1/2}\right)^T \qquad \Rightarrow \qquad \Gamma = \left(V\Lambda^{1/2}\right)^T, \tag{A.117}$$

where Λ is a diagonal matrix of eigenvalues and $\Lambda^{1/2}$ is then the diagonal matrix, taking the square root of each diagonal entry.

If P is symmetric, positive-definite, then the matrix square root can be computed much more efficiently using the Cholesky decomposition (Appendix A.7.3).

Square root matrices find three important uses:

1. *As the implicit representation of a positive-semidefinite matrix:*
   It can be numerically difficult to determine whether a given matrix is positive-semidefinite, and similarly difficult to guarantee that a numerical computation results in a positive-semidefinite matrix, although the consequences of inadvertently losing positive-semidefiniteness can be striking (divergent algorithms, negative error variances).

   Instead, if a matrix P (usually a covariance, either a prior model or an estimation error posterior) is represented by its square root

   $$P = \Gamma^T \Gamma, \tag{A.118}$$

   then *any* manipulation of $\Gamma \Rightarrow \bar{\Gamma}$, possibly including numerical rounding, yields a valid square root matrix $\bar{\Gamma}$ where

   $$\bar{P} = \bar{\Gamma}^T \bar{\Gamma} \geq \mathbf{0} \tag{A.119}$$

   is guaranteed to be positive-semidefinite.

2. *As a numerically-robust representation:*
   The condition number $\kappa(P)$ of a matrix P is a measure of conditioning, or numerical sensitivity. Given the eigendecomposition

   $$P\underline{v}_i = \lambda_i \underline{v}_i \qquad \text{or} \qquad PV = V\Lambda, \tag{A.120}$$

   if P is symmetric, positive-definite[2] then

   $$\kappa(P) = \frac{\max_i\{\lambda_i\}}{\min_i\{\lambda_i\}} \qquad \kappa(\sqrt{P}) = \kappa(\Gamma) = \kappa(V\Lambda^{1/2}) = \frac{\max_i\{\sqrt{\lambda_i}\}}{\min_i\{\sqrt{\lambda_i}\}}, \tag{A.121}$$

   therefore

   $$\kappa(\sqrt{P}) = \sqrt{\kappa(P)}, \tag{A.122}$$

   $$\log_{10} \kappa(\sqrt{P}) = \tfrac{1}{2} \log_{10} \sqrt{\kappa(P)} \tag{A.123}$$

   implying that the square root form requires only *half* the number of floating-point digits for an adequate, implicit representation of P.

3. *In random sampling:*
   If a random vector $\underline{x} \sim (\underline{\mu}, P)$ has covariance P, then given the covariance square root

   $$\bar{P} = \bar{\Gamma}^T \bar{\Gamma} \tag{A.124}$$

   we can find a random sample of $\underline{x}$ as

   $$\underline{x} = \underline{\mu} + \Gamma^T \underline{w} \qquad \underline{w} \sim I. \tag{A.125}$$

[2] Implying that eigenvalues and singular values are equal, allowing the discussion to be simplified.

A.9 Pseudoinverses

Rectangular matrices do not, by definition, possess an inverse. However, it is possible to define pairs of rectangular matrices which satisfy certain aspects of inversion.

In general, given a matrix A we might refer to its *pseudoinverse* as that matrix A^+ such that the product is as close as possible to the identity [141]:

$$A^+ = \arg_X \min \|AX - I\|_F = \arg_X \min \sum_{i,j} (AX - I)_{i,j}^2 , \tag{A.126}$$

a definition which is too vague and impractical, in general.

The most common definition of a pseudoinverse asserts the Moore–Penrose conditions [4, 141]:

$$A^+A = (A^+A)^T, \quad AA^+ = (AA^+)^T, \quad AA^+A = A, \quad A^+AA^+ = A^+ \tag{A.127}$$

implying that the repeated application of a pseudoinverse pair (AA^+) or (A^+A) leaves a product unchanged.

If A has full rank, often encountered in inverse problems, then the pseudoinverse is computable in closed form:

$$\begin{array}{ll} \text{Full Row Rank} & \Rightarrow \text{Nu}(A^T) = \{\underline{0}\} \Rightarrow A^+ = A^T(AA^T)^{-1} \Rightarrow AA^+ = I \\ \text{Full Column Rank} \Rightarrow & \text{Nu}(A) = \{\underline{0}\} \Rightarrow A^+ = (A^TA)^{-1}A^T \Rightarrow A^+A = I. \end{array} \tag{A.128}$$

The above definitions do lead to the sensible conclusion that the pseudoinverse matrix for an invertible matrix is, in fact, just the regular matrix inverse:

$$A \text{ invertible } \Rightarrow (A^TA)^{-1}A^T = A^T(AA^T)^{-1} = A^{-1}. \tag{A.129}$$

Pseudoinverses are used in reductions of dimensionality, such as in Section 8.2.2, and also in the solution to the least-squares problems of Chapter 3. In particular, suppose that

$$\underline{m} = C\underline{z} + \underline{v} \qquad E[\underline{v}] = \underline{0}, \ \ \text{cov}(\underline{v}) = I \tag{A.130}$$

where C is any real matrix. Then the least-squares minimum-norm solution for $\underline{z}$ is given by [4]

$$\hat{\underline{z}} = C^+\underline{m}, \tag{A.131}$$

valid whether C has full-row or full-column rank. That is, for a full-rank $k \times n$ matrix C:

$$\begin{array}{lllll} k = n & \Rightarrow \text{ Nonsingular} & \Rightarrow \hat{\underline{z}} = C^+\underline{m} = C^{-1}\underline{m} & & \text{Unique} \\ k < n & \Rightarrow \text{ Underdetermined} & \Rightarrow \hat{\underline{z}} = C^+\underline{m} = C^T(CC^T)^{-1}\underline{m} & & \text{Min. Norm} \\ k > n & \Rightarrow \text{ Overdetermined} & \Rightarrow \hat{\underline{z}} = C^+\underline{m} = (C^TC)^{-1}C^T\underline{m} & & \text{Least Squares} \end{array} \tag{A.132}$$

B

Statistics

This appendix provides a brief summary of univariate and multivariate statistics, covariances, and simple transformations of random variables. For a more detailed review the reader is referred to [37, 76, 99, 248, 284].

B.1 Random Variables, Random Vectors, and Random Fields

This section provides an overview of the quantities fundamental to this text, starting from random scalars, to random vectors, and then to random multidimensional fields.

B.1.1 Random Variables

A random variable is a single scalar which is random. Random variables are typically either discrete (a random integer, for example) or continuous (a random real number).

The nature of a random variable is characterized by its associated cumulative distribution function

$$F_x(\tau) = \Pr(x < \tau). \tag{B.1}$$

Typically more convenient for continuous random variables is the probability density function (PDF) $p_x()$, which satisfies

$$F_x(\tau) = \Pr(x < \tau) = \int_{-\infty}^{\tau} p_x(s)\, ds. \tag{B.2}$$

Only on occasion, when it is necessary to distinguish between a random variable and its particular instance, do we explicitly specify the PDF subscript; normally it will be understood from the context.

P. Fieguth, *Statistical Image Processing and Multidimensional Modeling*, Information Science and Statistics, DOI 10.1007/978-1-4419-7294-1_13, © Springer Science+Business Media, LLC 2011

The *expectation* operation is defined as

$$E\big[f(x)\big] = \int_{-\infty}^{\infty} f(x)p(x)\,dx, \tag{B.3}$$

where f is some mathematical function. Two very important choices of f lead to the definitions of mean and variance:

$$\begin{aligned} &\text{Mean:} && \mu_x = E\big[x\big] \\ &\text{Variance:} && \sigma_x^2 = E\big[x^2\big] - E\big[x\big]^2 = E\big[(x - E[x])^2\big]\ . \end{aligned}$$

The square root of the variance, σ_x, is referred to as the standard deviation of x.

Given independent samples $\check{x}_i$ of random variable x, we can estimate the sample statistics

$$\begin{aligned} &\text{Sample Mean:} && \hat{\mu}_x = \tfrac{1}{N}\textstyle\sum_{i=1}^{N} \check{x}_i \\ &\text{Sample Variance:} && \hat{\sigma}_x^2 = \tfrac{1}{N-1}\textstyle\sum_{i=1}^{N} \big(\check{x}_i - \hat{\mu}_x\big)^2\ . \end{aligned}$$

By far the most important and common PDF is the *Normal* or *Gaussian* distribution:

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma_x} e^{-\frac{1}{2}\left(\frac{x-\mu_x}{\sigma_x}\right)^2}. \tag{B.4}$$

B.1.2 Joint Statistics

A *joint* probability distribution $p(x, y)$ characterizes the relationship of two random variables x and y:

$$\Pr(x < \alpha, y < \beta) = \int_{-\infty}^{\alpha}\int_{-\infty}^{\beta} p(x, y)\,dx\,dy. \tag{B.5}$$

A *marginal* distribution is derived from a joint one by integrating out one or more variables; for example

$$p(x) = \int_{-\infty}^{\infty} p(x, y)\,dy. \tag{B.6}$$

Expectations for multiple variables are defined similarly to (B.3):

$$E\big[f(x, y)\big] = \int\int_{-\infty}^{\infty} f(x, y)p(x, y)\,dx\,dy. \tag{B.7}$$

We say that x and y are *independent* if

$$p(x, y) = p(x)p(y), \tag{B.8}$$

in which case knowing x tells us nothing about y. We say that x and y are *uncorrelated* if

$$E[xy] = E[x]E[y]. \tag{B.9}$$

If x and y are uncorrelated then they are not *linearly* related in any way, but *may* still be related in some *nonlinear* fashion. Independence implies uncorrelatedness, but not the other way around.

The correlation between x and y is defined as

$$E\big[(x - \mu_x)(y - \mu_y)\big], \tag{B.10}$$

which reduces to the simple $E[xy]$ if x and y have a mean of zero. Frequently more useful is a normalized version of the correlation, the correlation coefficient between x and y:

$$\rho_{x,y} = \frac{E\big[(x - \mu_x)(y - \mu_y)\big]}{\sigma_x \sigma_y}. \tag{B.11}$$

The correlation coefficient measures the ability to predict y as a linear function of x; $\rho_{x,y} = 0$ implies no predictability (x and y uncorrelated), and $\rho_{x,y} = \pm 1$ implies perfect predictability (x and y deterministically linearly related). It is always true that $|\rho(x, y)| \leq 1$.

We can generalize the above by introducing conditional statistics:

$p(x|y)$ is the PDF for x conditioned on another random variable y,
$p(x|A)$ is the PDF for x given that the event A took place.

Joint, marginal, and conditional densities are related by Bayes' rule:

$$p(x|y) = \frac{p(x, y)}{p(y)} = \frac{p(y|x)\, p(x)}{p(y)}, \tag{B.12}$$

which applies to continuous and/or discrete random variables, thus

$$\Pr(x|y)p(y) = p(y|x) \Pr(x) \tag{B.13}$$

for discrete x and continuous y.

Finally we can also define conditional expectations, consistent with our previous definition:

$$E\big[f(x)|y\big] = \int f(x)p(x|y)\, dx. \tag{B.14}$$

Note that we're integrating only over x, *not* over y; in (B.14) y is just a given piece of information, not a random variable. The variable of integration may be emphasized by writing the expectation as

$$E_x\big[f(x)|y\big]. \tag{B.15}$$

B.1.3 Random Vectors

The extension of random variables to random vectors is straightforward: a random vector $\underline{x}$ of dimension n is a column-vector of n random variables:

$$\underline{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \tag{B.16}$$

The PDF of $\underline{x}$ is the joint density function of its components:

$$\Pr(x_1 < \tau_1, \ldots, x_n < \tau_n) = \int_{-\infty}^{\tau_1} \cdots \int_{-\infty}^{\tau_n} p(\underline{x})\, d\underline{x}. \tag{B.17}$$

Note, however, that although $\underline{x}$ is now a vector, $p(\underline{x})$ is still a *scalar*!

Computing the joint density of a subset of the components of $\underline{x}$ is accomplished by computing the appropriate marginal distribution; for example

$$p(x_1, \ldots, x_{l-1}, x_{l+1}, \ldots, x_n) = \int_{-\infty}^{\infty} p(\underline{x})\, dx_l. \tag{B.18}$$

Expectations of functions of random vectors are computed as before:

$$E\big[f(\underline{x})\big] = \int \cdots \int f(\underline{x}) p(\underline{x})\, d\underline{x} \tag{B.19}$$

which leads to the vectorized definitions

Mean: $\underline{\mu}_x = E\big[\underline{x}\big]$
Covariance: $\Sigma_x = E\big[(\underline{x} - \underline{\mu})(\underline{x} - \underline{\mu})^T\big]$

and the corresponding definitions for the sample statistics

Sample Mean: $\hat{\underline{\mu}}_x = \frac{1}{N} \sum_{i=1}^{N} \check{\underline{x}}_i$
Sample Covariance: $\hat{\Sigma}_x = \frac{1}{N-1} \sum_{i=1}^{N} \big(\check{\underline{x}}_i - \hat{\underline{\mu}}_x\big)\big(\check{\underline{x}}_i - \hat{\underline{\mu}}_x\big)^T$.

The covariance (also see Appendix B.4) is an n by n matrix, with an important structure: the (i, j)th entry in the matrix

$$(\Sigma_x)_{i,j} = E\big[(x_i - \mu_i)(x_j - \mu_j)\big] \tag{B.20}$$

is the correlation between x_i and x_j. Thus zero entries in Σ_x imply a decorrelation of the corresponding two variables.

We can also talk about the relationship *between* random vectors:

Cross-Covariance:	$\Sigma_{xy} = E\left[(\underline{x} - \underline{\mu}_x)(\underline{y} - \underline{\mu}_y)^T\right]$
Uncorrelated:	$E[\underline{x}\,\underline{y}^T] = E[\underline{x}]E[\underline{y}^T] \;\Rightarrow\; \Sigma_{xy} = \mathbf{0}$
Independent:	$p(\underline{x}, \underline{y}) = p(\underline{x})p(\underline{y})$

As before, independence implies uncorrelatedness, but not conversely.

Finally, Bayes' rule applies:

$$p(\underline{x}|\underline{y}) = \frac{p(\underline{x}, \underline{y})}{p(\underline{y})} = \frac{p(\underline{y}|\underline{x})p(\underline{x})}{p(\underline{y})}. \tag{B.21}$$

B.1.4 Random Fields

A random field [2, 62, 112, 335] $\underline{x}$ is a collection of random variables arranged on a lattice Ω:

$$X = \left\{ x_{\underline{i}} \in \Psi \mid \underline{i} \in \Omega \right\}. \tag{B.22}$$

In principle the lattice can be any (possibly irregular) collection of discrete points in any number of dimensions; however it is most convenient and intuitive to visualize the lattice as a rectangular, regular array of sites:

$$\Omega = \left\{ (i, j) \,|\, 1 \le i \le n_1, 1 \le j \le n_2 \right\} \tag{B.23}$$

in which case a random field is just a set of random pixels

$$X = \left\{ x_{i,j} \,|\, (i, j) \in \Omega \right\}. \tag{B.24}$$

A random field is (spatially) stationary if its statistics are only a function of offset, and not of position, for example that

$$E\left[x_{i,j} x_{i+\delta, j+\kappa}\right] \equiv E\left[x_{0,0} x_{\delta,\kappa}\right] \quad \forall i, j. \tag{B.25}$$

Similarly, a time-dynamic random field

$$X(t) = \left\{ x(t)_{i,j} \,|\, (i, j) \in \Omega \right\} \tag{B.26}$$

is temporally stationary if the statistics are only a function of temporal offset, and not of absolute time:

$$E\left[x(t)_{i,j} x(s)_{a,b}\right] \equiv E\left[x(0)_{i,j} x(s - t)_{a,b}\right]. \tag{B.27}$$

We therefore have the separate, distinct concepts of time stationarity and spatial stationarity. It is possible for $X(t)$ to have neither, one, or both forms of stationarity. It is similarly possible for a multidimensional field X to be stationary in one direction (e.g., along the columns) but not in another (the rows).

As with random variables or random vectors, any random field can, in principle, be completely characterized by its associated probability measure $p_X(X)$. The detailed form of $p(\cdot)$ depends on whether the alphabet Ψ of the elements $x_{i,j} \in \Psi$ is discrete, in which case $p_X(X)$ denotes a probability distribution, or continuous, in which case $p_X(X)$ denotes a probability density function.

Because any random field X can be lexicographically reordered into a column vector $\underline{x} = [X]_:$, all of the properties of random vectors and their associated covariances extend to random fields.

In most cases, the distinct feature of random fields is their size. Consider, for example, a modestly sized image, in which $n_1 = n_2 = 256$. In this case, X contains 65 536 random elements, and the joint distribution $p(\cdot)$ or the covariance $\mathrm{cov}([X]_:)$ must explicitly characterize the joint statistics of 65 536 elements.

Because the function $p(\cdot)$ is a cumbersome and computationally inefficient means of defining the statistics of a random field, a great part of the research into random fields involves the discovery or definition of *implicit* statistical forms which lead to effective or faithful representations of the true statistics, while admitting computationally efficient algorithms. Chapter 5 examines this question of representation at length.

B.2 Transformation of Random Vectors

It is common to operate on a random vector with some function

$$\underline{y} = \underline{f}(\underline{x}), \tag{B.28}$$

or, written in component form,

$$y_i = f_i(x_1, \ldots, x_n). \tag{B.29}$$

If $f()$ is a *linear* function then

$$\underline{y} = \underline{f}(\underline{x}) = A\underline{x} + \underline{b} \tag{B.30}$$

for some constant matrix A and vector $\underline{b}$; in this linear case the statistics of the transformation can be calculated in closed form:

$$\begin{aligned} \underline{\mu}_y &= E[\underline{y}] \\ &= E[A\underline{x} + \underline{b}] = AE[\underline{x}] + \underline{b} = A\underline{\mu}_x + \underline{b} \end{aligned} \tag{B.31}$$

$$\begin{aligned} \Lambda_y &= E\left[(\underline{y} - \underline{\mu}_y)(\underline{y} - \underline{\mu}_y)^T\right] \\ &= E\left[(A\underline{x} + \underline{b} - A\underline{\mu}_x - \underline{b})(A\underline{x} + \underline{b} - A\underline{\mu}_x - \underline{b})^T\right] \\ &= E\left[A(\underline{x} - \underline{\mu}_x)(\underline{x} - \underline{\mu}_x)^T A^T\right] \\ &= AE\left[(\underline{x} - \underline{\mu}_x)(\underline{x} - \underline{\mu}_x)^T\right] A^T \\ &= A\Lambda_x A^T. \end{aligned} \tag{B.32}$$

The shape of the PDF is generally distorted, even by linear transformations, except in one special case: linear transformations map normal distributions to normal distributions:

$$\underline{x} \sim \mathcal{N}(\underline{\mu}, \Sigma) \xrightarrow{\underline{y}=A\underline{x}+\underline{b}} \underline{y} \sim \mathcal{N}(A\underline{\mu} + \underline{b}, A\Sigma A^T). \tag{B.33}$$

In the case where the transformation is nonlinear the computation of the probability density is more difficult. Suppose we have random variables x, y, related as

$$y = f(x) \tag{B.34}$$

where f is continuous. The PDF of y is

$$p_y(Y) = \lim_{\delta \to 0} \frac{\Pr\big((Y-\delta) \le y \le (Y+\delta)\big)}{2\delta}. \tag{B.35}$$

That is, we're interested in determining the probability of finding y in some small window about Y. But y will be near Y only if x is near a root $f(X_i) - Y = 0$. Because f is continuous,

$$\lim_{\delta \to 0} f(X_i + \delta) = Y + \delta f'(X_i), \tag{B.36}$$

then given the q roots $X_1, \ldots, X_q$,

$$\Pr(|Y - y| \le \delta) = \sum_{i=1}^{q} \Pr\big(|X_i - x| \le \delta / f'(X_i)\big). \tag{B.37}$$

where the number q of roots will generally be a function of Y. Finally, taking the limit as $\delta \to 0$, we find

$$p_y(Y) = \frac{p_x(X_1)}{|f'(X_1)|} + \cdots + \frac{p_x(X_q)}{|f'(X_q)|} \tag{B.38}$$

as the nonlinear transformation from x to y.

B.3 Multivariate Gaussian Distribution

The common assumption of the Gaussian distribution is motivated on a variety of counts:

Common in the physical world due to the Central Limit Theorem,
Distribution preservation under linear transformations,
The equivalence of independence and uncorrelatedness,
The equivalence of the MAP and Bayesian least-squares estimators,
The linearity of the optimum Bayesian least-squares estimator.

The definition of the multivariate Gaussian is straightforward:

$$p(\underline{x}) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{1/2}} \exp\left[-\frac{(\underline{x}-\underline{\mu})^T \Sigma^{-1}(\underline{x}-\underline{\mu})}{2}\right]. \tag{B.39}$$

We briefly consider two special cases.

CASE 1: The dimension $n = 1$:

Setting $n = 1$ causes (B.39) to reduce to

$$p(x) = \frac{1}{(2\pi)^{\frac{1}{2}}|\Sigma|^{1/2}} \exp\left[-\frac{(x-\mu)^2\Sigma^{-1}}{2}\right] \tag{B.40}$$

$$= \frac{1}{(2\pi)^{\frac{1}{2}}\sigma} \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right] \tag{B.41}$$

where $\Sigma = \sigma^2$. That is, the multivariate Gaussian (B.39) properly reduces to the usual univariate Gaussian (B.4).

CASE 2: The covariance is diagonal:

Therefore the covariance matrix has the form

$$\Sigma = \begin{bmatrix} \sigma_{11}^2 & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \sigma_{nn}^2 \end{bmatrix} \tag{B.42}$$

The diagonality of Σ implies a simple form for the matrix inverse and determinant, thus

$$p(\underline{x}) = \frac{1}{(2\pi)^{\frac{n}{2}}\left(\prod_1^n \sigma_{ii}^2\right)^{\frac{1}{2}}} \exp\left[-\frac{1}{2}\sum_{i=1}^n \left(\frac{x_i-\mu_i}{\sigma_{ii}}\right)^2\right] \tag{B.43}$$

$$= \prod_{i=1}^n \frac{1}{(2\pi)^{\frac{1}{2}}\sigma_{ii}} \exp\left[-\frac{1}{2}\left(\frac{x_i-\mu_i}{\sigma_{ii}}\right)^2\right] \tag{B.44}$$

$$= \prod_{i=1}^n p(x_i). \tag{B.45}$$

That is, if $\underline{x}$ is a multivariate Gaussian, and if the elements of $\underline{x}$ are uncorrelated, then the elements of $\underline{x}$ are also independent.

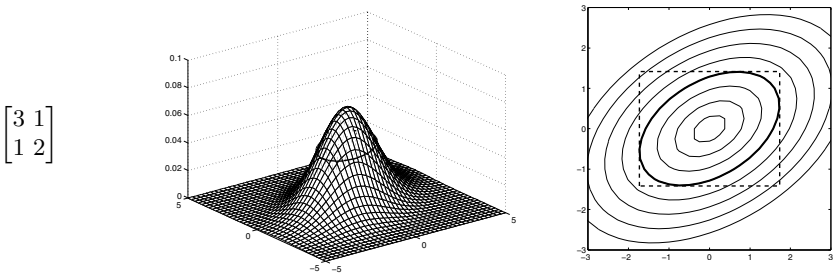


Fig. B.1. A two-dimensional normal distribution is characterized by its two-by-two covariance matrix, left. By slicing through the distribution, middle, the contour of constant probability is seen to be an ellipse. The constant-probability contours are most easily seen in a contour plot, right, where the thick line shows the *unit* standard deviation contour.

B.4 Covariance Matrices

A covariance matrix

$$\Sigma_x = E\left[(\underline{x} - \underline{\mu})(\underline{x} - \underline{\mu})^T\right] \tag{B.46}$$

describes the second-order interrelationships of the elements of a random vector $\underline{x}$. A single element of the covariance

$$(\Sigma_x)_{i,j} = E\big[(x_i - \mu_i)(x_j - \mu_j)\big] \tag{B.47}$$

is the correlation between x_i and x_j. Thus zero entries in Σ_x imply a decorrelation of the corresponding two variables, and a diagonal covariance implies that *all* of the variables are mutually decorrelated.

All covariances must satisfy four properties:

$\Sigma = \Sigma^T$	Symmetry,
$\Sigma_{i,i} \geq 0$	Non-negativity of diagonal elements,
$\Sigma \geq \mathbf{0}$	Positive-semidefiniteness (Appendix A.4), and
$\lambda_j \geq 0$	Non-negativity of eigenvalues (Appendix A.7.1).

The cross-covariance

$$\Sigma_{xy} = E\left[(\underline{x} - \underline{\mu}_x)(\underline{y} - \underline{\mu}_y)^T\right] \tag{B.48}$$

also allows us to describe the relationship *between* random vectors, however *cross*-covariances do not obey any of the symmetry, non-negativity, or positive-definiteness

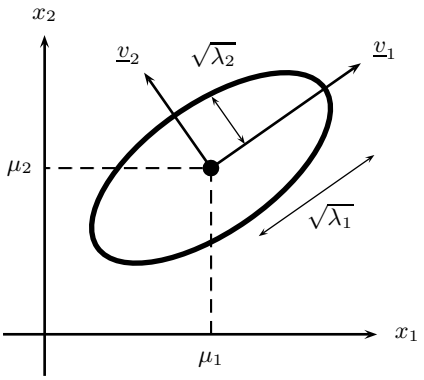


Fig. B.2. A covariance describes an n-dimensional ellipsoid, with the eigenvectors pointing in the direction of the principal axes, and the eigenvalues describing the axis lengths.

properties of covariances. However, it is true that $\Sigma_{xy} = \Sigma_{yx}^T$ and that a zero entry in ith column and jth row of Σ_{xy} implies a decorrelation between y_i and x_j.

Returning to the multivariate normal distribution of (B.39), the equiprobability contour (the collection of points in space, all of which are equally likely) is defined by

$$p(\underline{x}) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{1/2}} \mathrm{e}^{-\frac{1}{2}(\underline{x}-\underline{\mu})^T \Sigma^{-1}(\underline{x}-\underline{\mu})} = \text{constant}, \tag{B.49}$$

that is, that

$$(\underline{x}-\underline{\mu})^T \Sigma^{-1}(\underline{x}-\underline{\mu}) = \text{constant}. \tag{B.50}$$

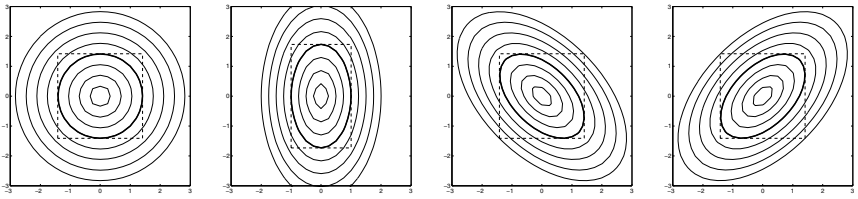
The set of points described by (B.50) forms an ellipse, as seen in Figure B.1. In general the behaviour of a covariance is sketched by drawing the *unit* standard deviation contour, where

$$(\underline{x}-\underline{\mu})^T \Sigma^{-1}(\underline{x}-\underline{\mu}) = 1. \tag{B.51}$$

This unit-ellipse is centred on the mean $\underline{\mu}$, has axes pointing in the directions of the eigenvectors of Σ, and has semi-axis lengths equal to the square roots of the eigenvalues of Σ. The relationships between a covariance and its associated eigendecomposition are summarized in Figure B.2. The matrix properties of covariances are further discussed in the context of eigendecompositions in Appendix A.7.1, specifically on page 399. Certain properties of *cross*-covariances are discussed in the context of the singular value decomposition in Appendix A.7.2.

In general, for the bivariate case a covariance is written as

$$\Sigma = \begin{bmatrix} a & b \\ b & c \end{bmatrix} \qquad a, c \geq 0 \quad |b| \leq \sqrt{ac}, \tag{B.52}$$



$\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$ $\begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix}$ $\begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}$ $\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$

Uncorrelated, Equal Variances | Uncorrelated, Unequal Variances | Negatively Correlated | Positively Correlated

Fig. B.3. Four examples of two-dimensional Normal distributions. In each case, the darkly banded contour shows the unit standard deviation distance, and the dashed rectangle plots the bounding box.

where the inequality constraint on b ensures that the covariance remains positive-definite. The unit standard deviation ellipse associated with Σ is perfectly inscribed in a rectangle, centred on the mean, of width $2\sqrt{a}$ and height $2\sqrt{c}$, where the angle and eccentricity of the ellipse are controlled by the sign and magnitude of b. Four examples are plotted in Figure B.3.

C

Image Processing

Although this text is not about image processing per se, some familiarity with common concepts in image processing, such as convolution or denoising, is very helpful. This appendix is only a brief list of concepts, and is in no way a comprehensive tutorial on image processing, for which the interested reader is referred to any one of many excellent textbooks [36, 54, 143, 174, 210].

An image I is a set of values arranged on a rectangular grid. An image may be considered a two-dimensional function

$$I(x, y)$$

or as a matrix

$$I_{i,j}$$

with the frustrating disadvantage that the spatial indexing of the two notational schemes are quite different from each other, as illustrated in Figure C.1.

Any image stored in a computer is an approximate representation of a real-world phenomenon:

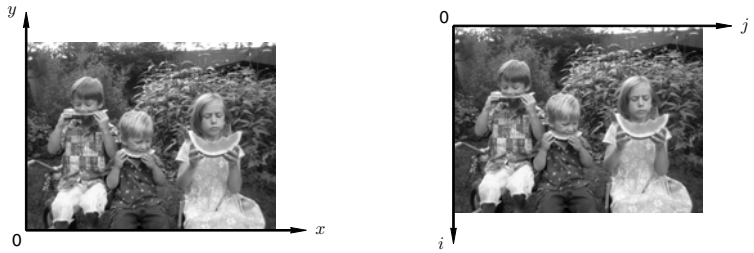
Real-World Image:	$x,\ y,\ I(x,y)$	are all continuous-valued
Computer-Stored Image:	$x,\ y,\ I(x,y)$	are all discretized.

As this text concerns the computer processing of multidimensional data, we focus on the latter definition.

For most images, $I(x, y)$ is either a scalar, in which case the image is referred to as "greyscale," or $\underline{I}(x, y)$ is a vector, in which case we have a colour image in some colour space. By far the most common colour space is RGB,

$$\underline{I}(x, y) = \begin{bmatrix} R(x, y) \\ G(x, y) \\ B(x, y) \end{bmatrix} \tag{C.1}$$

P. Fieguth, *Statistical Image Processing and Multidimensional Modeling*, Information Science and Statistics, DOI 10.1007/978-1-4419-7294-1_14, © Springer Science+Business Media, LLC 2011



Usual Cartesian Coordinate System $I(x, y)$ Usual Matrix Indexing System I_{ij}

Fig. C.1. Two coordinate systems are commonly used in image processing.

such that each image pixel consists of red, green, and blue values. Many other colour spaces have been defined (HSV, YIQ, ...), but are not relevant to this text. Instead, in this text the value of an image $I(x, y)$ can be any unknown or measured quantity, such as the temperature of the ocean, the strength of a radar return, or the rate of signal decay in an MRI.

C.1 Convolution

In signal processing, *any* linear operation on a signal can be written in terms of what is known as a convolution [244]

$$s(t) * h(t) = \int_{-\infty}^{\infty} s(\tau)h(t-\tau)\,d\tau \qquad s(n) * h(n) = \sum_{\tau=-\infty}^{\infty} s(\tau)h(n-\tau), \quad \text{(C.2)}$$

where both the continuous-time and discrete-time versions are given, respectively. h is known as the impulse response, and characterizes the linear operation.

Precisely the same is true in two (and higher) dimensions, such that

$$\begin{aligned} \bar{I}(x, y) = I(x, y) * H(x, y) &= \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} I(m, n)H(x-m, y-n) \\ &= \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} I(x-m, y-n)H(m, n), \end{aligned} \quad \text{(C.3)}$$

where H, typically known as a *convolution kernel*, describes a pattern or a mask such that a single element in the result $\bar{I}$ is formed as a weighted sum of elements in I, where the kernel H consists of a matrix of weights.

Consider three illustrations:

IMAGE BLURRING:
In an optical system $\mathcal{H}$, say consisting of one or more lenses, the *point-spread-function* H is the image that results from a point (impulsive) light source

$$H = \mathcal{H}\big(\delta(x,y)\big).$$

If the system is linear, such that superposition applies,

$$\mathcal{H}\big(\alpha I_1 + \beta I_2\big) \equiv \alpha\mathcal{H}\big(I_1\big) + \beta\mathcal{H}\big(I_2\big), \tag{C.4}$$

and stationary, meaning that a shift in the input leads to a corresponding shift in the output,

$$\bar{I}(x,y) = \mathcal{H}\big(I(x,y)\big) \implies \bar{I}(x-\Delta_x, y-\Delta_y) = \mathcal{H}\big(I(x-\Delta_x, y-\Delta_y)\big), \tag{C.5}$$

then the system can be written as a convolution, with the point-spread-function (the "impulse response") as the convolution kernel:

$$\bar{I} = \mathcal{H}\big(I\big) = H * I. \tag{C.6}$$

Two examples of blurring are shown in Figure C.2. There is no particular need for H to be isotropic, as is illustrated by the anisotropic example in the figure.

EDGE DETECTION:
If an edge is defined as an abrupt change in image brightness, then a differencing operator can be used to reveal edges.

Among the many edge detectors which have been proposed, two of the simplest are the Sobel operators S_x, S_y shown in the bottom panel of Figure C.2. Many other variations are possible, including diagonal differences (rather than just horizontal and vertical), and spatially-adaptive thresholds on the gradient.

IMAGE FILTERING:
In general, a convolution is just the filtering of an image. The blur and edge illustrations in Figure C.2 are examples of low-pass and high-pass filtering, respectively.

Methods of filter design have been developed (see any of the image processing textbooks cited at the beginning of this appendix) which allow standard one-dimensional filters, such as notch-pass or band-pass filters, to be generalized to two dimensions.

The convolution equations in (C.2) are very elegant in theory, but become considerably more complicated in practice because images are *finite*: they have boundaries.

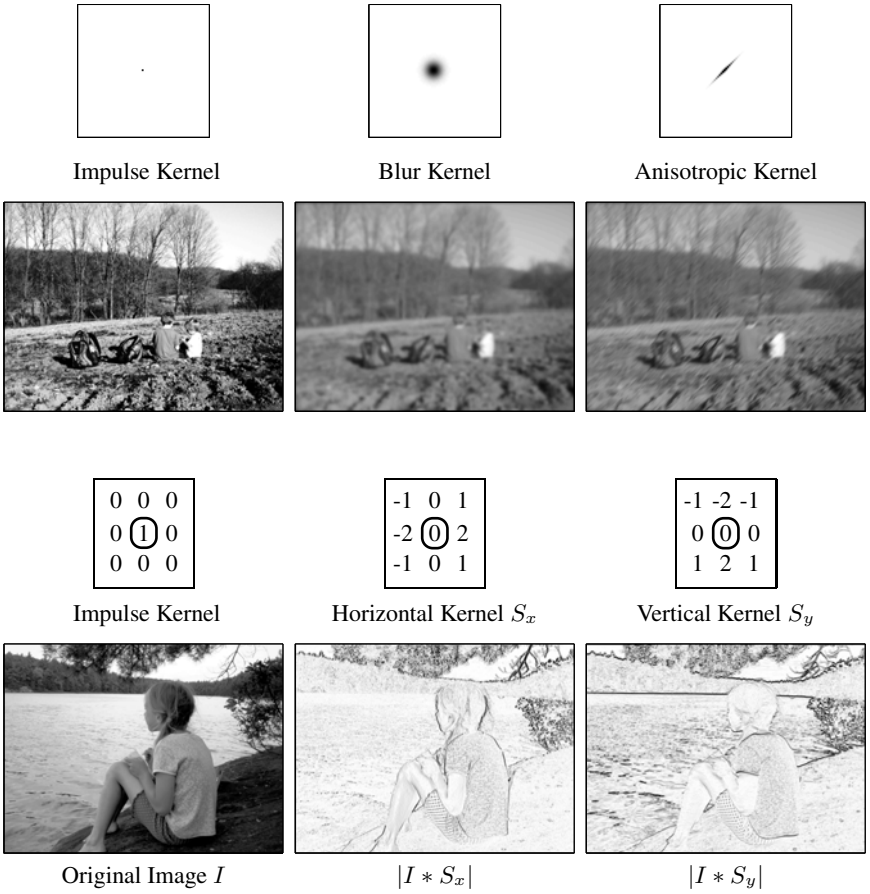
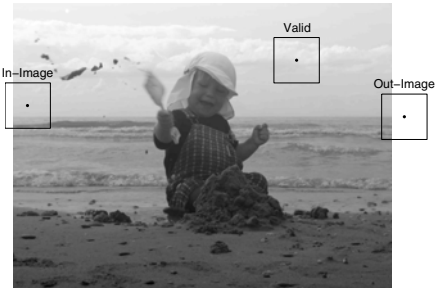


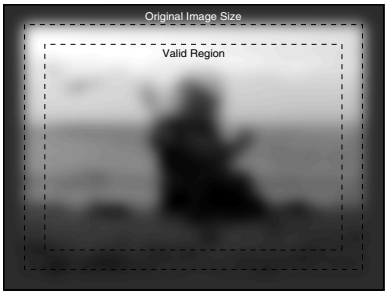
Fig. C.2. Convolution is a powerful concept in image processing. An impulsive kernel is the identity operator, left, causing no change to the image. Other convolution kernels may cause blurring, top, or detect edges, bottom.

We have a few different options, illustrated in Figure C.3:

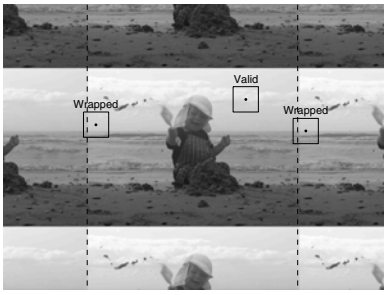
1. Compute the convolution only *within* the image, away from the boundary. This approach corresponds to the "Valid Region" in Figure C.3.
2. Just truncate the convolution sum outside of the image, which implicitly assumes the image to be *zero* outside of its boundaries. This approach corresponds to keeping the dark boundary of $I * H$ in Figure C.3.



Original Image I

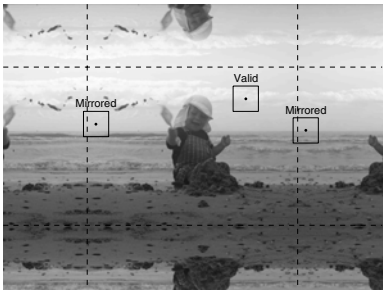


Convolved Image $I * H$



Periodic Image I

Circular Convolved Image $I \circledast H$



Mirrored Image I

Convolved Mirrored Image

Fig. C.3. The behaviour of convolution is somewhat subtle at image boundaries. For regular convolution over a finite image, top, the image is implicitly treated as zero outside of its domain, leading to the boundary effect seen in $I * H$. The convolution is unaffected by the boundary only within the "Valid" region. Under circular convolution, middle, the image is implicitly treated as periodic, which can lead to boundary effects if the periodic assumption is a poor one (observe the darkening near the top of the circular convolution, due to wrap-around from the dark bottom). Mirroring the image at its boundaries frequently leads to better boundary behaviour.

3. Assume the image to be periodic, and therefore nonzero outside of its boundary, leading to what is known as *circular convolution*. If the image is not, in fact, periodic then spill-over effects may be visible, as can be seen in the top and bottom of $I \circledast H$ in Figure C.3.

4. Mirror the image at its boundaries, rather than assume it to be periodic. This approach turns out to have a higher computational complexity than the simpler periodic assumption, but tends to lead to better results for real images.

C.2 Image Transforms

Images can be large, highly complex, with foregrounds and backgrounds and structures on a variety of scales. In order to be able to analyze the information content of an image more simply, many transformations have been proposed for one purpose or another. The transformations may be colour or greyscale, linear or nonlinear, local or global, flat or hierarchical, a change of basis or a dimensionality reduction.

Three particularly common transforms are shown in Figure C.4:

THE FOURIER TRANSFORM [244] represents an image as a weighted sum of sinusoids, analogous to the one-dimensional Fourier transform. The transform is global and assumes the image to be periodic, limiting the usefulness and applicability of this transformation when processing real images. However, the existence of a very fast algorithm, the *Fast Fourier Transform*, makes the transform of great interest for certain problems in statistical image processing (see Sections 8.3 and 11.2.1).

The Fourier transform has a very close connection to circular convolution and linear systems. For any image I and impulse response H,

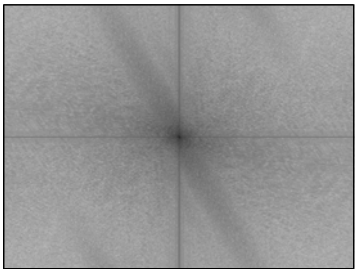
$$\mathrm{FFT}(I \circledast H) = \mathrm{FFT}(I) \odot \mathrm{FFT}(H) \tag{C.7}$$

That is, circular convolution corresponds to element-by-element multiplication in the Fourier domain.

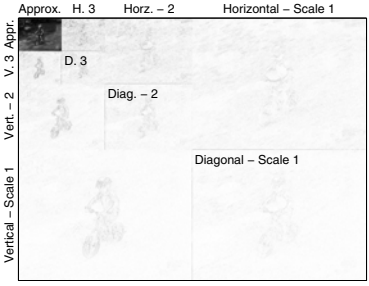
THE WAVELET TRANSFORM [293] is a local, hierarchical transform, far more effective than the Fourier transform in processing real images. The image is represented as a weighted sum of shifted and rescaled versions of a single function (the *wavelet*). In many cases the wavelet transform is very sparse (most coefficients are near zero), making the approach very successful in image compression and denoising. The wavelet transform can be used as a preconditioner for spatial problems, as discussed in Section 8.4.2.

THE HOUGH TRANSFORM [54] searches an image for lines. Whereas the Fourier and wavelet transforms are *invertible*, meaning that the image can be recovered

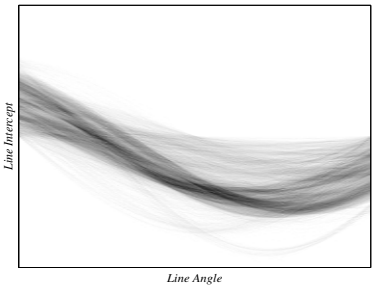
Original Image



Fourier Transform



Wavelet Transform



Hough Transform

Fig. C.4. A great many image transformations have been proposed. Observe particularly the sparsity of the wavelet transform, and the diagonal band in the Fourier transform stemming from the camera motion tracking the bicyclist in the original image. The Hough transform was applied to the Vertical-Scale 1 image from the wavelet transform.

from the transform coefficients, the Hough transform is a method of image analysis, not one of representation. For every straight line, parametrized by its slope and intercept, the Hough transform sums the image along that line, leading to peaks in the transform corresponding to detected lines in the image.

Clearly the Hough transform can be generalized to other parametrized shapes, such as circles or parabolae.

Other important transformations, described in image processing texts, include Gabor filters, Laplacian pyramids, difference of Gaussians, and the Cosine transform.

C.3 Image Operations

Many algorithms and methods have been developed for image processing. A very short, incomplete list follows:

DENOISING: Given an image corrupted by noise, estimate a noise-reduced image. The noise may be possibly additive or multiplicative, possibly Gaussian or non-Gaussian, possibly white or correlated (Figure C.5 and Figure 2.3).

IN-PAINTING: A generalization of the denoising problem, given an image with missing pieces, develop a way to extrapolate the observed behaviour of the image into the missing parts (Figure 2.3).

DEBLURRING OR DECONVOLUTION: Given an image corrupted by some point-spread-function, invert the point-spread convolution to obtain the original image. The point-spread-function may be known or, if unknown, the problem is referred to as *blind* deconvolution (Figure 2.3).

RESOLUTION ENHANCEMENT OR SUPERRESOLUTION: Given one or more images at some resolution, estimate the image at a higher resolution (Application 11 and Section 8.4.3).

EDGE DETECTION: Find the edges / lines in an image, often an initial operation to simplify image analysis for further image segmentation, feature detection, or object recognition (Figure C.6).

CLASSIFICATION: Classify each image pixel into one of k possible predetermined behaviours. A great many satellite remote-sensing problems fall into this category, distinguishing urban and rural, ice and water, forest and agricultural etc. (Section 7.1.3 and Application 6).

SEGMENTATION: Essentially the blind version of image classification, dividing an image into non-overlapping regions of homogeneous behaviour, but where the number of regions and their meaning are unknown ahead of time (Figure C.7 and Application 7).

COMPRESSION: Find a transformation whereby an image can be represented, and subsequently reconstructed, from as few coefficients as possible (Section 8.2.1).

FEATURE DETECTION: Extract significant points of interest from an image, often corners. Preferably the features are robust, invariant to rotation, scale, or changes in illumination.

OBJECT RECOGNITION: Similar to classification, produce a map which identifies objects of interest in an image (car, person, ...) as distinct from the background.

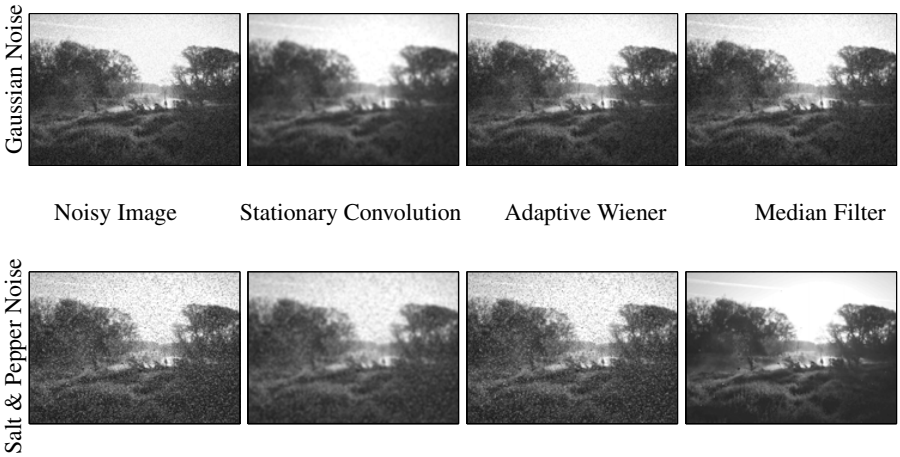


Fig. C.5. A small sampling of the many denoising methods which have been developed. A regular convolution leads to excessive smoothing, however the Wiener filter is least-squares optimal for Gaussian noise, and the nonlinear median filter performs very well for the non-Gaussian salt-and-pepper noise.

WATERMARKING: Via subtle changes in certain image features, place into the image a digital signature or watermark, in a way that is robust to image rotation, rescaling, cropping etc.

TRACKING AND REGISTRATION: Associate the objects / features in one image with those in one or more other images.

Of these operations, three have a somewhat greater relevance to this text:

1. Image denoising is an inverse problem, as discussed in Chapter 2. Almost all noise-reduction methods proceed on the basis of assuming that neighbouring pixels in an image are closely related and can be averaged, such as using a simple convolutional blur, as shown in Figure C.5.

   Convolution is indiscriminate, however, blurring all parts of an image equally. More refined approaches are adaptive, blurring more broadly in smooth regions, and more locally in regions having greater detail. Nonlinear methods, such as a median filter, attempt to preserve edge structure by not averaging across an edge.

   Some of the more powerful denoising approaches (see Problem 8.5) involve nonlinear processing in the sparse wavelet domain.

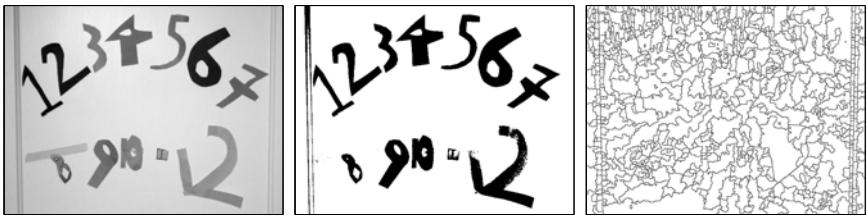
Original Image Sobel Edges Canny Edges

Fig. C.6. The Sobel edge detector is based on the simple convolutions in Figure C.2. The more sophisticated Canny detector is among the most widely-used approaches.

2. The simple Sobel convolutional operators demonstrated in Figure C.2 are really a very primitive approach to edge detection. A well-established, more reliable approach to edge detection is the Canny detector, shown in Figure C.6. An alternative approach is the Zero-Cross method, based on finding zero-crossings in the second derivative of an image.

3. Image segmentation seeks to divide an image into homogeneous regions. Essentially, image segmentation is a *dual* to edge detection, in the sense that the outlines of the segmented regions offer one possible edge map for an image. The resulting segmentation is a hidden label map (per Chapter 7).

   The simplest approach to segmentation is a global, single threshold ζ, such that the image is divided as shown in Figure C.7. Clearly non-global variations can be proposed, such that threshold $\zeta(x, y)$ varies with location. There are exceptionally many approaches to segmentation, including clustering methods based on K-means [91], region growing / region merging methods, active contours (snakes), level-set methods, graph-based methods, scale-space methods, and watershed.



Original Image Otsu Global Thresholding Watershed Segmentation

Fig. C.7. An image can be segmented into pieces by binarization with a single global threshold, such as using Otsu's method. Most complex images do not threshold well with a single threshold, so local methods, such as watershed, are commonly used.

Reference Summary

Long bibliographic lists can be difficult to use, therefore this section attempts to provide some thematic structure to assist the reader in finding meaningful references and further reading.

Because textbooks and journal papers fulfill rather different purposes in terms of depth, breadth, accessibility, and on-line availability, the textbook and paper references are listed separately.

Classic Papers:

3, 19, 25, 27, 85, 86, 127, 136, 171, 175, 182, 221, 223, 250, 265, 266, 303, 308, 320, 332, 337

Exceptionally Cited Papers:

More than 5000 Citations:

32, 37, 76, 85, 91, 122, 127, 141, 162, 165, 174, 179, 185, 212, 221, 224, 243, 248, 249, 264, 265, 283, 308

More than 1000 Citations:

2, 8, 19, 23, 25, 26, 28, 29, 38, 43, 45, 55, 73, 77, 83, 87, 88, 93, 94, 98, 99, 125, 131, 134, 139, 143, 145, 147, 148, 152, 158, 160, 163, 170, 175, 177, 180, 181, 183, 195, 198, 204, 205, 210, 222, 223, 233, 244, 250, 253, 269, 277, 278, 281, 287, 293, 294, 296, 297, 302, 313, 319, 320, 324

Inverse Problems:

Texts 12, 24, 49, 154, 298, 301, 310, 313
Papers 27, 129, 139, 140, 172, 173, 189, 304, 308, 309, 322

Statistical Methods:

Kalman Filtering:
Texts 7, 8, 17, 28, 32, 47, 97, 101, 125, 151, 156, 231, 272, 287, 326
Papers 10, 11, 16, 20, 41, 65–69, 74, 100, 103, 121, 123, 164, 181, 182, 188, 191, 192, 239, 266, 269, 307, 311, 325, 333, 337, 338

Estimation:
Texts 8, 17, 125, 298
Papers 18, 70, 75, 81, 82, 102, 105, 106, 108–110, 114, 123, 135, 145, 155, 169, 178, 191, 194, 213, 214, 219, 227, 232, 237, 299, 333, 334, 345, 349

General Statistics:
Texts 2, 6, 23, 37, 55, 64, 76, 99, 101, 137, 150, 151, 166, 248, 252, 261, 270, 271, 281, 284, 290, 324
Papers 3, 5, 25, 27, 48, 57, 71, 75, 85, 93, 100, 114, 115, 117, 136, 139, 140, 164, 171, 175, 179, 182, 184, 220, 223, 227, 242, 250, 256, 262, 263, 266, 276, 283, 297, 311, 312, 319, 323, 344

Hierarchical Methods:

Wavelets:
Texts 293, 317
Papers 13, 56, 57, 87, 114, 115, 117, 149, 178, 221, 222, 237, 262, 263, 274, 286, 306, 339, 340, 345

Hierarchical Statistics:
Texts 317
Papers 13, 18, 35, 58, 70, 81, 82, 102, 105, 106, 108–110, 112, 144, 146, 149, 169, 186, 191, 197, 199, 213–215, 232, 234–237, 299, 306, 329, 334, 341, 345

Other Hierarchical:
Texts 40, 43, 92, 152, 153, 224, 228, 249, 291, 330
Papers 18, 38, 44, 50, 53, 68–70, 79, 81, 82, 102, 105, 106, 108–110, 169, 190, 191, 214, 229, 232, 251, 273, 275, 279, 299, 303, 316, 331, 332

Image Processing:

Markov Random Fields:
Texts 2, 42, 207, 335, 336
Papers 26,35,58–62,71,86,112,126,127,129,136,146,147,157,186,196–199, 213, 225, 226, 234–236, 245, 246, 254, 255, 276, 292, 295, 327, 334, 341

Hidden/Discrete Models:
Texts 52, 96, 137, 177, 180, 205, 335
Papers 19,21,22,77,85,116,147,184,193,206,230,260,265,274,275,314,320

Image/Spatial Statistics:
Texts 55, 76, 101, 270, 271, 290, 291, 328
Papers 5, 25, 65, 66, 75, 100, 164, 171, 184, 192, 220, 241, 242, 258, 297, 311, 312, 329, 337, 338

Other:
Texts 30, 32, 36, 45, 54, 92, 143, 174, 183, 210, 259
Papers 9, 16, 31, 33, 51, 84, 87, 111, 118, 124, 161, 167, 170, 185, 188, 202, 214, 216–218, 251, 253, 267, 288, 296, 300, 302–305, 331

Applications:

Remote Sensing:
10, 53, 67, 72, 80, 81, 105, 109, 111, 120, 121, 123, 128, 134, 135, 138, 158, 187, 191, 203, 204, 219, 232, 238, 240, 282, 342, 343

Object Tracking:
16, 17, 31, 33, 36, 74, 84, 107, 124, 170, 269, 272, 305, 318

Computer Vision:
15, 30, 32, 36, 94, 95, 107, 119, 159–161, 167, 178, 185, 201, 207, 209, 251, 253, 256, 267, 288, 296, 298–300, 303, 305, 307, 349

Medical Imaging:
36, 51, 183, 218, 220, 319

Other Applications:
113, 148, 208, 236, 291, 332

References

1. J. Abbott, M. Bronstein, T. Mulders, "Fast deterministic computation of determinants of dense matrices," *Int. Conf. Symbolic and Algebraic Computation*, Vancouver, pp.197–204, 1999
2. R. Adler, *The Geometry of Random Fields*, Wiley, 1981
3. H. Akaike, "Markovian representation of stochastic processes by canonical variables," *SIAM J. Control* (13) #1, pp.162–173, 1975
4. A. Albert, *Regression and the Moore-Penrose Pseudoinverse*, Academic Press, 1972
5. S. Alexander, P. Fieguth, M. Ioannidis, E. Vrscay, "Hierarchical annealing for synthesis of binary porous media images," *Mathematical Geosciences* (41) #4, pp.357–378, 2009
6. T. Anderson, *The Statistical Analysis of Time Series*, Wiley, 1971
7. D. Angwin, H. Kaufman, *Digital Image Restoration* Springer, 1991
8. B. Anderson, J. Moore, *Optimal Filtering*, Prentice-Hall, 1979
9. G. Arce, "Multistage Order Statistic filters for image sequence processing," *IEEE Trans. Acoustics, Speech, Signal Processing* (39) #5, pp.1147–1163, 1991
10. A. Asif, J. Moura, "Data assimilation in large time varying multidimensional fields," *IEEE Trans. Image Processing* (8) #11, pp.1593–1607, 1999
11. A. Asif, J. F. Moura, "Block matrices with L-Block banded inverse: inversion algorithms," *IEEE Trans. Signal Processing* (53) #2, pp.630–642, 2005
12. R. Aster, B. Borchers, C. Thurber, *Parameter Estimation and Inverse Problems*, Academic Press, 2005
13. Z. Azimifar, P. Fieguth, E. Jernigan, "Towards random field modeling of wavelet statistics," *ICIP'02*, Rochester, 2002
14. S. Baker, T. Kanade, "Limits on super-resolution and how to break them," *IEEE CVPR*, 2000
15. Ballard, Hinton, Sejnowski, "Parallel computation in vision problems," *Nature* (306) #5938, pp.21–26, 1983
16. F. Barbaresco, S. Bonney, J. Lambert, B. Monnier, "Motion-based segmentation and tracking of dynamic radar clutter," *ICIP'96* (III), pp.923–926, 1996
17. Y. Bar Shalom, T. Fortmann, *Tracking and Data Association,* Academic Press, 1988
18. M. Basseville, A. Benveniste, K. Chou, S. Golden, R. Nikoukhah, A. Willsky, "Modeling and estimations of multiresolution stochastic processes," *IEEE Trans. Information Theory* (38) #2, pp.766–784, 1992

19. K. Baum, T. Petrie, G. Soules, N. Weiss, "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains," *The Annals of Mathematical Statistics* (41) #1, pp.164–171, 1970
20. M. Bello, A. Willsky, B. Levy, "Construction and applications of discrete-time smoothing error models," *Int. J. Control* (50) #1, pp.203–223, 1989
21. D. Benboudjema, W. Pieczynski, "Unsupervised statistical segmentation of nonstationary images using triplet Markov fields," *IEEE Trans. PAMI* (29) #8, pp.1367–1378, 2007
22. C. Benedekand, T. Sziranyiand, Z. Kato, J. Zerubia, "A multi-layer mrf model for object-motion detection in unregistered airborne image-pairs," *IEEE ICIP* (VI), pp.141–144, 2007
23. J. Beran, *Statistics for Long-Memory Processes*, Chapman & Hall, 1994
24. M. Bertero, P. Boccacci, *Introduction to Inverse Problems in Imaging*, Taylor & Francis, 1998
25. J. Besag, "Spatial interaction and the statistical analysis of lattice systems," *J. Royal Society, Series E* (36), pp.192–236, 1974
26. J. Besag, "On the statistical analysis of dirty pictures," *J. Royal Statistatical Society B* (48) #3, pp.256–302, 1986
27. M. Bertero, T. Poggio, V. Torre, "Ill-posed problems in early vision," *Proc. IEEE* (76) #8, pp.869–889, 1988
28. G. Bierman, *Factorization Methods for Discrete Sequential Estimation,* Academic Press, 1977
29. C. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, 1995
30. A. Blake, A. Yuille (Eds.), *Active Vision*, MIT Press, 1993
31. A. Blake, R. Curwen, A. Zisserman, "A framework for spatiotemporal control in the tracking of visual contours," *Int. J. Computer Vision* (11) #2, pp.127–145, 1993
32. A. Blake, *Active Contours*, Springer, 1998
33. S. Blostein, T. Huang, "Detecting small, moving objects in image sequences using sequential hypothesis testing," *IEEE Signal Processing* (39) #7, pp.1611–29, 1991
34. L. Blum, F. Cucker, M. Shub, S. Smale, *Complexity and Real Computation*, Springer, 1997
35. C. A. Bouman and M. Shapiro, "A multiscale random field model for Bayesian image segmentation," *IEEE Trans. Image Processing,* (3) #2, pp.162–177, March 1994
36. A. Bovik (Ed.), *Handbook of Image and Video Processing, 2nd ed.*, Academic Press, 2005
37. G. Box, G. Jenkins, G. Reinsel, *Time Series Analysis - Forecasting and Control*, Prentice-Hall, 1994,
38. A. Brandt, "Multi-level adaptive solutions to boundary-value problems," *Mathematics of Computation* (31) #138, pp.333–390, 1977
39. J. Bramble, J. Pasciak, A. Schatz, "The construction of preconditioners for elliptic problems by substructuring III," *Mathematics of Computation* (51) #184, pp.415–430, 1988
40. J. Bramble, *Multigrid Methods*, Wiley, 1993
41. J. Brailean, R. Kleihorst, S. Efstratiadis, A. Katsaggelos, A. Lagendijk, "Noise reduction filters for dynamic image sequences: A review," *Proc. IEEE* (83) #9, pp.1272–1292, 1995
42. P. Bremaud, *Markov Chains: Gibbs Fields, Monte Carlo Simulation, and Queues*, Springer, 1999
43. W. Briggs, *A Multigrid Tutorial*, SIAM, 1987
44. W. Briggs, "Wavelets and multigrid," *SIAM J. Scientific Computing* (14), 1993
45. P. Brodatz, *Textures: A Photographic Album for Artists and Designers,* Dover, 1966

46. M. Brookes, *The Matrix Reference Manual*, Imperial College, 2005
47. R. Bucy, P. Joseph, *Filtering for Stochastic Processes*, Wiley, 1968
48. C. Byrnes, S. Gusev, A. Lindquist, "A convex optimization approach to the rational covariance extension problem," *SIAM J. Control and Optimization* (37) #1, pp.211–229, 1999
49. S. Campbell, C. Meyer, *Generalized Inverses of Linear Transformations,* Dover, 1991
50. W. Campaigne, P. Fieguth, S. Alexander, "Frozen-state hierarchical annealing," *ICIAR'06 (Springer LNCS 4141)*, 2006
51. E. Candès, J. Romberg, T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Trans. Information Theory* (52) #2, pp.489–509, 2006
52. O. Cappé, E. Moulines, T. Ryden, *Inference in Hidden Markov Models*, Springer, 2005
53. G. Carballo, P. Fieguth, "Multiresolution network flow phase unwrapping," *IEEE Trans. Geoscience and Remote Sensing* (40) #8, pp.1695–1708, 2002
54. K. Castleman, *Digital Image Processing*, Prentice-Hall, 1996
55. D. Chandler, *Introduction to Modern Statistical Mechanics*, Oxford University Press, 1987
56. S. Chang, B. Yu, M. Vetterli, "Adaptive wavelet thresholding for image denoising and compression," *IEEE Trans. Image Processing* (9) #9, pp.1532–1546, 2000
57. S. Chang, Y. Bin, M. Vetterli, "Spatially adaptive wavelet thresholding with context modeling for image denoising," *IEEE Trans. Image Processing* (9) #9, pp.1532–1546, 2000
58. P. Charbonnier, L. Blanc-Feraud, M. Barlaud, "Noisy image restoration using multiresolution Markov random fields," *J. Visual Communication and Image Representation* (3) #4, pp.338–346, 1992
59. R. Chellappa, R. Kashyap, "Digital image restoration using spatial interaction models," *IEEE Trans. Acoustics, Speech, Signal Processing* (30) #3, pp.461–472, 1982
60. R. Chellappa and S. Chatterjee, "Classification of textures using Gaussian Markov random fields." *IEEE Trans. Acoustics, Speech, Signal Processing*, (33), pp.959–963, 1985
61. R. Chellappa, "Two-dimensional discrete Gaussian Markov random field models for image processing," *Progress in Pattern Recognition* (2), pp.79–112, 1985
62. R. Chellappa, A. Jain (Eds.), *Markov Random Fields – Theory and Application*, Academic Press, 1993
63. K. Chen, *Matrix Preconditioning Techniques and Applications*, Cambridge University Press, 2005
64. M. Chen, Q. Shao, J. Ibrahim, *Monte Carlo Methods in Bayesian Computation*, Springer Series in Statistics, Springer, 2000
65. T. Chin, W. C. Karl, A. Willsky, "Sequential filtering for multi-frame visual reconstruction," *Signal Processing* (28), pp.311–333, 1992
66. T. M. Chin, W. C. Karl, A. S. Willsky, "A distributed and iterative method for square root filtering in space-time estimation," *Automatica* (31) #1, pp.67–82, 1995
67. T. M. Chin, A. J. Mariano, and E. P. Chassignet, "Spatial regression and multiscale approximations for sequential data assimilation in ocean models," *J. Geophysical Research* (104), pp.7991–8014, 1999
68. K. Chou, *A Stochastic Modeling Approach to Multiscale Signal Processing*, PhD Thesis, Dept. EECS, Massachusetts Institute of Technology, 1991
69. K. Chou, A. Willsky, A. Benveniste, "Multiscale recursive estimation, data fusion, and regularization," *IEEE Trans. Automatic Control* (39) #3, pp.464–478, 1994
70. S. Clippingdale, R. Wilson, "Least-squares image estimation on a multiresolution pyramid," *ICASSP'89*, Glasgow, pp.1409–1412, 1989

71. J. Coleman, *Gaussian Spacetime Models: Markov Field Properties*, PhD Thesis, University of California at Davis, 1995
72. M. Costantini, "A novel phase unwrapping method based on network programming," *IEEE Trans. Geoscience and Remote Sensing* (36) #3, pp.813–821, 1998
73. R. Courant, D. Hilbert, *Methods of Mathematical Physics V1*, Interscience Publishers, 1953
74. I. Cox, S. Hingorani, "An efficient implementation of Reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking," *IEEE Trans. PAMI* (18) #2, pp.138–150, 1996
75. N. Cressie, "The origins of kriging," *Math. Geol.* (22) #3, pp.239–252, 1990
76. N. Cressie, *Statistics for Spatial Data*, Wiley, 1993
77. M. Crouse, R. Nowak, R. Baraniuk, "Wavelet-based statistical signal processing using hidden Markov models," *IEEE Trans. Signal Processing* (46) #4, pp.886–902, 1998
78. Dahlquist, Bjorck, *Numerical Methods*, Prentice-Hall, 1974
79. W. Dahmen, A. Kunoth, "Multilevel preconditioning," *Numerische Mathematik* (63) #3, pp.315–344, 1992
80. R. Daley, *Atmospheric Data Analysis*, Cambridge University Press, 1991
81. M. Daniel, A. Willsky, "A multiresolution methodology for signal-level fusion and data assimilation with applications to remote sensing," *Proc. IEEE*, (85), pp.164–180, 1997
82. M. Daniel, A. Willsky, "The modeling and estimation of statistically self-similar processes in a multiresolution framework," *IEEE Trans. Information Theory* (45) #3 , pp.955–970, 1999
83. P. Davis, *Circulant Matrices*, Wiley-Interscience, 1979
84. J. Davis, A. Bobick, "The representation and recognition of human movement using temporal templates," *CVPR*, pp.928–934, Puerto Rico, 1997
85. A. Dempster, N. Laird, D. Rubin, "Maximum likelihood estimation from incomplete data," *J. Royal Statistical Society (B)* (39) #1, pp.1–38, 1977
86. H. Derin, P. Kelly, "Discrete-index Markov-type random processes," *Proc. IEEE* (77) #10, pp.1485–1510, 1989
87. D. Donoho, I. Johnstone, "Adapting to unknown smoothness via wavelet shrinkage," *J. Am. Statistical Assocation* (90), pp.1200–1224, 1995
88. A. Doucet, N. de Freitas, N. Gordon (Eds.), *Sequential Monte Carlo Methods in Practice*, Springer, 2001
89. J. Driscoll, D. Healy, "Computing Fourier transforms and convolutions on the 2-Sphere," *Adv. in Appl. Math.* (15), pp.202–250, 1994
90. I. Drori, D. Cohen-Or, H. Yeshurun, "Fragment-based image completion," *ACM Trans. Graphics* (22) #3, pp.303–312, 2003
91. R. Duda, P. Hart, D. Stork, *Pattern Classification*, Wiley, 2001
92. D. Dudgeon, R. Mersereau, *Multidimensional Digital Signal Pocessing*, Prentice-Hall, 1984
93. B. Efron, G. Gong, "A leisurely look at the bootstrap, the jacknife, and cross-validation," *The American Statistician*, 1983
94. A. Efros, T. Leung, "Texture synthesis by non-parametric sampling," *IEEE ICCV*, 1999
95. A. Eleftheriadis, A. Jacquin, "Automatic face location detection and tracking for model-assisted coding of video teleconference sequences at low bit rates," *Signal Processing – Image Communication* (7) #3, pp.231–248, 1995
96. R. Elliot, L. Aggoun, J. Moore, *Hidden Markov Models: Estimation and Control, 3rd ed.*, Springer, 2008
97. R. Eubank, *A Kalman Filter Primer*, CRC Press, 2006

98. D. Evans, *Preconditioning Methods: Analysis and Application*, Gordon & Breach, 1983
99. Evans, Hastings, and Peacock, *Statistical Distributions*, Institute of Physics Publishers, 2001
100. G. Evensen, "Sequential data assimilation with nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics," *J. Geophysical Research* 99 (C5), pp.143–162, 1994
101. G. Evensen, *Data Assimilation: The Ensemble Kalman Filter*, Springer, 2007
102. E. Fabre, "New fast smoothers for multiscale systems," *IEEE Trans. Signal Processing* (44) #8, pp.1893–1911, 1996
103. B. Farrell, P. Ioannou, "State estimation using a reduced-order Kalman filter," *J. Atmospheric Sciences* (58), pp.3666–3680, 2001
104. S. Farsiu, M. Robinson, M. Elad, P. Milanfar, "Fast and robust multiframe super resolution," *IEEE Trans. Image Processing* (13) #10, pp.1327–1344, 2004
105. P. Fieguth, W. Karl, A. Willsky, C. Wunsch, "Multiresolution optimal interpolation and statistical analysis of TOPEX/POSEIDON satellite altimetry," *IEEE Trans. Geoscience and Remote Sensing* (33) #2, pp.280–292, 1995
106. P. W. Fieguth, A. S. Willsky, "Fractal estimation using models on multiscale trees," *IEEE Trans. Signal Processing* (44) #5, pp.1297–1300, 1996
107. P. Fieguth, Demetri Terzopoulos, "Color-based tracking of heads and other mobile objects at video frame rates," *CVPR'97*, pp.21–28, Puerto Rico, 1997
108. P. Fieguth, W. Karl, A. Willsky, "Efficient multiresolution counterparts to variational methods for surface reconstruction," *Computer Vision & Image Understanding* (70) #2, pp.157–176, 1998
109. P. Fieguth, D. Menemenlis, T. Ho, A. Willsky, C. Wunsch, "Mapping Mediterranean altimeter data with a multiresolution optimal interpolation algorithm," *J. Atmospheric and Oceanic Technology* (15), pp.535–546, 1998
110. P. Fieguth, "Multiply-rooted multiscale models for large-scale estimation," *IEEE Image Processing* (10) #11, pp.1676–1686, 2001
111. P. Fieguth, D. Menemenlis, I. Fukumori, "Mapping and pseudo-inverse algorithms for ocean data assimilation," *IEEE Trans. Geoscience and Remote Sensing* (41) #1, pp.43–51, 2003
112. P. Fieguth, J. Zhang, "Random field models" in A. Bovik (Ed.), *Handbook of Image and Video Processing* , Academic Press, p.361-376, 2005
113. W. Fieguth, *Multi-Input Quasi-Linearization,* Ph.D. Thesis, Dept. of Electrical Engineering, University of New Brunswick, 1967
114. M. Figueiredo, R. Nowak, "Wavelet-based image estimation: An empirical Bayes approach using Jeffreys' noninformative prior," *IEEE Trans. Image Processing* (10) #9, pp.1322–1331, 2001
115. M. Figueiredo, R. Nowak, "An EM algorithm for wavelet-based image restoration," *IEEE Trans. Image Processing* (12) #8, pp.906–916, 2003
116. S. Fine, Y. Singer, N. Tishby, "The hierarchical hidden markov model: analysis and applications," *Machine Learning* (32), pp.41–62, 1998
117. P. Flandrin, "Wavelet analysis and synthesis of fractional Brownian motion," *IEEE Trans. Information Theory* (38) #2, pp.910–917, 1992
118. R. Franktot, R. Chellappa, "A method for enforcing integrability in shape from shading algorithms," *IEEE Trans. PAMI* (10) #4, pp.439–451, 1989
119. W. Freeman, T. Jones, E. Pasztor, "Example-based super-resolution," *IEEE Computer Graphics and Applications*, pp.56–65, 2002

120. L. Fu, E. Christensen, C. Yamarone, M. Lefebvre, Y. Menard, M. Dorrer, P. Escudier, "TOPEX/POSEIDON mission overview," *J. Geophysical Research* (99) #C12, pp.24369–24381, 1994
121. I. Fukumori, P. Malanotte-Rizzoli, "An approximate Kalman filter for ocean data assimilation; an example with an idealized Gulf Stream model," *J. Geophysical Research*, 1994
122. K. Fukunaga, *Introduction to Statistical Pattern Recognition*, Academic Press, 1990
123. P. Gaspar, C. Wunsch, "Estimates from altimeter data of baryotropic Rossby waves in the northwestern Atlantic ocean," *J. Physical Oceanography* (19) #12, pp.1821–1844, 1989
124. D. Geiger, A. Gupta, L. Costa, J. Vlontzos, "Dynamic programming for detecting, tracking, and matching deformable contours," *IEEE Trans. PAMI* (17) #3, pp.294–302, 1995
125. A. Gelb, *Applied Optimal Estimation*, MIT Press, 2002
126. S. Gelfand, S. Mitter, "On sampling methods and annealing algorithms" in R. Chellappa, A. Jain (Eds.), *Markov Random Fields – Theory and Application*, pp.499–515, 1993
127. S. Geman, D. Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images," *IEEE Trans. PAMI* (6) #6, pp.721–741, 1984
128. D. Geman, J. Jedynak, "An active testing model for tracking roads in satellite images," *IEEE Trans. PAMI* (18) #1, pp.1–14, 1996
129. D. Geman, "Random fields and inverse problems in imaging," *Lecture Notes in Mathematics* (1427), Springer, pp.117–193, 1991
130. A. George, "Nested dissection of a regular finite element mesh," *SIAM J. Numerical Analysis*, pp.345–363, 1973
131. A. George, J. Liu, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall, 1981
132. A. George, M. Heath, J. Liu, E. Ng, "Sparse Cholesky factorizatoin on a local-memory multiprocessor," *Faculty of Mathematics Technical Report* CS-86-02, University of Waterloo, 1986
133. A. George, J. Gilbert, J. Liu (Eds), *Graph Theory and Sparse Matrix Computation*, Springer, 1993
134. D. Ghiglia, M. Pritt, *Two-Dimensional Phase Unwrapping*, Wiley, 1998
135. M. Ghil, P. Malanotti-Rizzoli, "Data assimilation in meteorology and oceanography," *Advances in Geophysics* (33), pp.141–266, 1991
136. B. Gidas, "A renormalization group approach to image processing problems," *IEEE Trans. PAMI* (11) #2, pp.164–180, 1989
137. W. Gilks, S. Richardson, D. Spiegelhalter (Eds), *Markov Chain Monte Carlo in Practice*, Chapman & Hall, 1996
138. R. Goldstein, H. Zebker, C. Werner, "Satellite radar interferometry — two-dimensional phase unwrapping," *Radio Science* (23) #4, pp.713–720, 1988
139. G. Golub, M. Heath, G. Wahba, "Generalized cross validation," *Technometries* (21), p.215, 1979
140. G. Golub, U. von Matt, "Generalized cross-validation for large scale problems," *J. Computational and Graphical Statistics* (6) #1, pp.1–34, 1997
141. G. Golub, C. Van Loan, *Matrix Computations*, Johns Hopkins University Press, 1996
142. G. Golub, D. O'Leary, "Some history of the conjugate gradient and Lanczos algorithms," *SIAM Review* (31), pp.50–102, 1989
143. R. Gonzalez, R. Woods, *Digital Image Processing (3rd ed.)*, Prentice-Hall, 2007
144. J. Goodman, A. Sokal, "Multigrid Monte Carlo method. conceptual foundations," *Physical Review D* (40) #6, pp.2035–2071, 1989

145. N. Gordon, D. Salmond, A. Smith, "Novel approach to nonlinear / nonGaussian Bayesian state estimation," *IEE Procedings* (F140), pp.107–113, 1993
146. C. Graffigne, F. Heitz, P. Perez, F. Prłteux, M. Sigelle, J. Zerubia, "Hierarchical Markov random field models applied to image analysis: a review," *SPIE* (2568), 1995
147. P. Green, "Reversible jump Markov chain Monte Carlo computation and Bayesian model determination," *Biometrika* (82) #4, p.711, 1995
148. L. Greengard, V. Rokhlin, "A fast algorithm for particle simulations," *J. Computational Physics* (73), pp.325–348, 1987
149. H. Greenspan, C. Anderson, S. Akber, "Image enhancement by nonlinear extrapolation in frequency space," *IEEE Trans. Image Processing* (9) #6, pp.1035–1047, 2000
150. U. Grenander, *Elements of pattern theory*, Johns Hopkins University Press, 1996
151. M. Grewal, A. Andrews, *Kalman Filtering: Theory and Practice*, Prentice-Hall, 1993
152. W. Hackbusch, *Multi-Grid Methods and Applications*, Springer, 1985
153. W. Hackbusch, *Iterative Solution of Large Sparse Systems of Equations*, Springer, 1994
154. J. Hadamard, *Lectures on the Cauchy Problem in Linear Partial Differential Equations,* Yale University Press, 1923
155. J. Handschin, "Monte Carlo techniques for prediction and filtering of non-linear stochastic processes," *Automatica* (6), pp.555–563, 1970
156. M. Hayes, *Statistical Digital Signal Processing and Modeling*, Wiley, 1996
157. X. He, R. Zemel, M. Carreira-Perpiná, "Multiscale conditional random fields for image labeling," *IEEE CVPR*, 2004
158. W. Heiskanen, H. Moritz, *Physical Geodesy*, W.H. Freemand & Co., 1967
159. B. Horn, M. Brooks, "Integrability of Surface Gradients," *MIT A.I. Memo 813*, 1985
160. B. Horn, *Robot Vision*, MIT Press, 1986
161. B. Horn, "Height and gradient from shading," *Int. J. Computer Vision* (5) #1, pp.37–46, 1990
162. R. Horn, C. Johnson, *Matrix Analysis*, Cambridge University Press, 1990
163. R. Horn, C. Johnson, *Topics in Matrix Analysis*, Cambridge University Press, 1994
164. P. Houtekamer, H. L. Mitchell, "Data assimilation using an ensemble Kalman filter technique," *Monthly Weather Review* (126), pp.796–811, 1998
165. A. Hyvärinen, E. Oja, "Independent component analysis: algorithms and applications," *Neural Networks* (13) #4-5, pp.411–430, 2000
166. A. Hyvärinen, J. Karhunen, E. Oja, *Independent Component Analysis*, Wiley, 2001
167. Ikeuchi, B. Horn, "Numerical shape from shading and occluding boundaries," *Artificial Intelligence* (17) #1-3, pp.141–184, 1981
168. I. Ipsen, C. Meyer, "The idea behind Krylov methods," *American Mathematical Monthly* (105) #10, pp.889–899, 1998
169. W. Irving, P. Fieguth, A. Willsky, "An overlapping tree approach to multiscale stochastic modeling and estimation," *IEEE Trans. Image Processing* (6) #11, pp.1517–1529, 1997
170. M. Isard, A. Blake, "CONDENSATION - conditional density propagation for visual tracking," *Int. J. Computer Vision* (29), pp.5–28, 1998
171. E. Ising, "Beitrag zur Theorie des Ferromagnetismus," *Z. Phys.* (31), pp.253–258, 1925
172. V. Ivanov, "On linear problems which are not well-posed," *Soviet Math. Dokl.* (3), pp.981–983, 1962
173. V. Ivanov, "The approximate solution of operator equations of the first kind," *USSR Copm. Math. Phys* (6), pp.197–205, 1966
174. A. Jain, *Fundamentals of Digital Image Processing*, Prentice-Hall, 1989
175. A. Jain, R. Duin, J. Mao, "Statistical pattern recognition: a review," *IEEE Trans. PAMI* (22) #1, pp.4–37, 2000

176. M. Jamieson, P. Fieguth, L. Lee, "Parametric contour estimation by simulated annealing," *IEEE ICIP*, Spain, 2003
177. F. Jensen, *Bayesian Networks and Decision Graphs*, Springer, 2001
178. F. Jin, P. Fieguth, L. Winger, "Wavelet video denoising with regularized multiresolution motion estimation," *EURASIP J. Applied Signal Processing* #72705, 2006
179. I. Jolliffe, *Principal Components Analysis (2nd ed.)*, Springer, 2002
180. M. Jordan, *Learning in graphical models*, Kluwer Academic Publishers, 1998
181. S. J. Julier, J. K. Uhlmann, "A new extension of the Kalman filter to nonlinear systems," *Proc. AeroSense*, 1997
182. R. Kalman, "Contributions to the theory of optimal control," *Bol. Soc. Mat. Mexicana* (5), pp.102–119, 1960
183. A. Kak, M. Slaney, *Principles of Computerized Tomographic Imaging*, IEEE, 1999
184. D. Kandel, E. Domany, "General cluster Monte Carlo dynamics," *Physical Review B* (43) #10, pp.8539–8548, 1991
185. M. Kass, A. Witkin, D. Terzopoulos, "Snakes: active contour models," *Int. J. Computer Vision* (1) #4, pp.321–331, 1988
186. Z. Kato, M. Berthod, J. Zerubia, "A hierarchical Markov random field model and multitemperature annealing for parallel image classification," *Graphical Models and Image Processing* (58) #1, pp.18–37, 1996
187. W. Kaula, *Theory of Satellite Geodesy*, Blaisdell Publishing Co., 1966
188. H. Kaufman, J. Woods, M. Tekalp, S. Dravida, "Estimation and identification of two-dimensional images," *IEEE Trans. Automatic Control* (AC-28), pp.745–756, 1983
189. D. Keren, M. Werman, "Probabilistic analysis of regularization," *IEEE Trans. PAMI* (15) #10, pp.982–995, 1993
190. M. Khalil, P. Wesseling, "Vertex-centered and cell-centered multigrid for interface problems," *Journal of Computational Physics* (98) #1, pp.1–10, 1992
191. F. Khellah, P. Fieguth, J. Murray, M. Allen, "Statistical processing of large image sequences," *IEEE Trans. Image Processing* (14) #1, pp.80–93, 2005
192. J. Kim, J. Woods, "Spatio-temporal adaptive 3D Kalman filter for video," *IEEE Trans. Image Processing* (6) #3, p.414, 1997
193. J. Kim, R. Zabih, "Factorial Markov random fields," *ECCV*, pp.321–334, 2002
194. G. Kitagawa, "Monte Carlo filter and smoother for non-Gaussian non-linear state space models," *J. Computational and Graphical Statistics* (5) #1, pp.1–25, 1996
195. B. Kosko (Ed) *Neural Networks for Signal Processing*, Prentice-Hall, pp.37–61, 1992
196. S. Kumar, M. Hebert, "Discriminative random fields: a discriminative framework for contextual interaction in classification," *Proc. IEEE ICCV* (2), pp.1150–1157, 2003
197. J.M. Laferte, P. Perez, F. Heitz, "Discrete Markov image modeling and inference on the quadtree," *IEEE Trans. Image Processing* (9) #3, pp.390–404, 2000
198. J. Lafferty, A. McCallum, F. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," *Proc. Int. Conf. on Machine Learning (ICML)*, pp.282–289, 2001
199. S. Lakshmanan, H. Derin, "Gaussian Markov random fields at multiple resolutions," in *Markov Random Fields – Theory and Application* (R. Chellappa, A. Jain (Eds)), pp.131–157, 1993
200. P. Lancaster, L. Rodman, *Algebraic Riccati Equations*, Oxford University Press, 1995
201. K. Lee, C. Kuo, "Shape from shading with a linear triangular element surface model," *IEEE Trans. PAMI* (15) #8, pp.815–822, 1993
202. D. Lee, J. Shiau, "Thin plate splines with discontinuities and fast algorithms for their computation," *SIAM J. Scientific Computing* (15) #6, pp.1311–1330, 1994

203. P. LeTraon, P. Gaspar, F. Bouyssel, H. Makhmara, "Using Topex/Poseidon data to enhance ERS-1 data," *J. Atmospheric and Oceanic Technology* (12), pp.161–170, 1995
204. S. Levitus, *Climatological Atlas of the World Ocean*, United States Government Printing, 1982
205. J. Li, R. Gray, *Image Segmentation and Compression Using Hidden Markov Models*, Kluwer Academic, 2000
206. J. Li, A. Najmi, R. Gray, "Image classification by a two dimensional hidden Markov model," *IEEE Trans. Signal Processing* (48) #2, pp.517–533, 2000
207. S. Li, *Markov Random Field Modeling in Computer Vision*, Springer, 2001
208. Z. Liang, C. Fernandes, F. Magnani, P. Philippi, "A reconstruction technique for three-dimensional porous media using image analysis and Fourier transforms," *J. Petroleum Science and Engineering* (21) #3-4, pp.273–283, 1998
209. L. Liang, C. Liu, Y. Xu, B. Guo, H. Shum, "Real-time texture synthesis by patch-based sampling," *ACM Trans. Graphics* (20) #3, pp.150, 2001
210. J. Lim, *Two-Dimensional Signal and Image Processing*, Prentice-Hall, 1990
211. J. Liu, "Computational models and task scheduling for parallel sparse Cholesky factorization," *Parallel Computing* (3), pp.327–342, 1986
212. L. Ljung, *System Identification : Theory for the User*, Prentice-Hall, 1987
213. M. Luettgen, W. Karl, A. Willsky, R. Tenney, "Multiscale representations of Markov random fields," *IEEE Trans. Signal Processing* (41) #12, pp.3377–3396, 1993
214. M. Luettgen, W. Karl, A. Willsky, "Efficient multiscale regularization with applications to the computation of optical flow," *IEEE Trans. Image Processing* (3) #1, pp.41–64, 1994
215. M. Luettgen, A. Willsky, "Multiscale smoothing error models," *IEEE Trans. Automatic Control* (40) #1, 1995
216. T. Lundahl, W. Ohley, S. Kay, R. Siffert, "Fractional Brownian motion: A maximum likelihood estimator and its application to image texture," *IEEE Trans. Medical Imaging* (5), pp.152–161, 1986
217. J. Luo, C. Guo, "Perceptual grouping of segmented regions in color images," *Pattern Recognition* (36), pp.2781–2792, 2003
218. M. Lustig, D. Donoho, J. Pauly, "Sparse MRI: The application of compressed sensing for rapid MR imaging," *Magnetic Resonance in Medicine* (58) #6, pp.1182–1195, 2007
219. P. Malanotte-Rizzoli, "Data assimilation: fundamentals, global and Mediterranean examples," in P. Malanotte-Rizzoli, A. Robinson (Eds.), *Ocean Processes in Climate Dynamics: Global and Mediterranean Examples* , NATO Asi Series (419), 1994
220. F. Maes, A. Collignon, D. Vandermeulen, G. Marchal, P. Suetens, "Multimodality image registration by maximization of mutual information," *IEEE Trans. Medical Imaging* (16) #2, pp.187–198, 1997
221. S. Mallat, "A theory of multiresolution signal decomposition: The wavelet representation," *IEEE Trans. PAMI* (11) #7, pp.674–693, 1989
222. S. Mallat, S. Zhong, "Characterization of signals from multiscale edges," *IEEE Trans. PAMI* (14) #9,pp.710–732, 1992
223. B. Mandelbrot, J. van Ness, "Fractional Brownian motions, fractional noises and applications," *SIAM Review* (10), pp.422–437, 1968
224. B. Mandelbrot, *The Fractal Geometry of Nature*, W.H. Freeman & Co., 1982
225. B. Manjunath, T. Simchony, R. Chellappa, "Stochastic and Deterministic Networks for Texture Segmentation," *IEEE Trans. Acoustics, Speech, Signal Processing* (38) #6, pp.1039–1049, 1990
226. B. Manjunath, R. Chellappa, "Unsupervised texture segmentation using Markov random field models," *IEEE Trans. PAMI* (13) #5, pp.478–482, 1991

227. G. Matheron, Les Variables Régionalisées et Leur Estimation, Masson, 1965
228. S. McCormick, *Multigrid methods*, SIAM, 1987
229. S. McCormick, *Multilevel Adaptive Methods for Partial Differential Equations*, SIAM, 1989
230. D. Melas, S. Wilson, "Double Markov random fields and Bayesian image segmentation," *IEEE Trans. Signal Processing* (50) #2, pp.357–365, 2002
231. J. Mendel, *Lessons in Estimation Theory for Signal Processing, Communications, and Control*, Prentice-Hall, 1995
232. D. Menemenlis, P. Fieguth, C. Wunsch, A. Willsky, "Adaptation of a fast optimal interpolation algorithm to the mapping of oceanographic data," *J/ Geophysical Research* (102) #C5, pp.10573–10584, 1997
233. C. Meyer, *Matrix Analysis and Applied Linear Algebra*, SIAM, 2001
234. M. Mignotte, C. Collet, P. Perez, P. Bouthemy, "Sonar image segmentation using an unsupervised hierarchical MRF model," *IEEE Trans. Image Processing* (9) #7, pp.1216–1231, 2000
235. M. Mignotte, "Nonparametric multiscale energy-based model and its application in some imagery problems," *IEEE Trans. PAMI* (26) #2, pp.184–197, 2004
236. A. Mohebi, P. Fieguth, M. Ioannidis, "Statistical fusion of two-scale images of porous media," *Advances in Water Resources* (32), pp.1567–1579, 2009
237. E. Simoncelli, P. Muller, B. Vidakovic (Eds), *Bayesian Inference in Wavelet-Based Methods*, Springer, 1999
238. W. Munk, P. Worcester, C. Wunsch, *Ocean Acoustic Tomography*, Cambridge University Press, 1995
239. K. Nagpal, R. Helmick, C. Sims, "Reduced-order estimation Part 1: Filtering," *Int. J. Control* (45) #6, pp.1867–1888, 1987
240. R. Nash, S. Jordan, "Statistical geodesy – an engineering perspective," *Proc. IEEE* (66) #5, pp.532–550, 1978
241. T. Ojala, M. Pietikinen, D. Harwood, "A comparative study of texture measures with classification based on feature distributions," *Pattern Recognition* (29), pp.51–59, 1996
242. T. Ojala, M. Pietikainen, T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Trans. PAMI* (24) #7, pp.971–987, 2002
243. A. Oppenheim, R. Schafer, *Discrete-time signal processing (3rd ed.)*, Prentice-Hall, 2009
244. A. Oppenheim, A. Willsky, H. Nawab, *Signals & Systems*, Prentice-Hall, 1997
245. M. Ortner, X. Descombes, J. Zerubia, "Building outline extraction from digital elevation models using marked point processes," *Int. J. Computer Vision* (72) #2, pp.107–132, 2007
246. M. Ortner, X. Descombes, J. Zerubia, "A marked point process of rectangles and segments for automatic analysis of digital elevation models," *IEEE Trans. PAMI* (30), pp.105–119, 2009
247. V. Pan, *How to Multiply Matrices Faster*, Springer, 1984
248. A. Papoulis, S. Pillai, *Probability, Random Variables, and Stochastic Processes*, McGraw Hill, 2002
249. J. Pearl, *Probabilistic Reasoning in Intelligent Systems*, Morgan Kaufmann Publishers, 1988
250. K. Pearson, "On lines and planes of closest fit to systems of points in space," *Philosophical Magazine* (2), pp.559–572, 1901
251. S. Peleg, G. Ron, "Nonlinear multiresolution: a shape from shading example," *IEEE Trans. PAMI* (12) #12, pp.1206–1210, 1990

252. D. Percival, A. Walden, *Spectral Analysis for Physical Applications*, Cambridge University Press, 1993
253. A. Pentland, B. Moghaddam, T. Starner, "View-based and modular eigenspaces for face recognition," *CVPR*, pp.84–91, 1994
254. P. Perez, F. Heitz, "Restriction of a Markov random field on a graph and multiresolution statistical image modeling," *IEEE Trans. Information Theory* (42)#1, pp.180–190, 1996
255. P. Perez, "Markov random fields and images," CWI Quarterly (11) #4, pp.413–437, 1998
256. P. Perez, J. Vermaak, A. Blake, "Data fusion for visual tracking with particles," *Proc. IEEE* (92) #3, pp.495–513, 2004
257. K. Petersen, M. Pedersen, *The Matrix Cookbook*, Technical University of Denmark, 2008
258. H. Permuter, J. Francos, I. Jermyn, "A study of Gaussian mixture models of color and texture features for image classication and segmentation," *Pattern Recognition* (39), pp.695–706, 2006
259. M. Petrou, P. Sevilla, *Dealing with Texture*, Wiley, 2006
260. W. Pieczynski, A. Tebbache, "Pairwise Markov random fields and segmentation of textured images," *Machine Graphics & Vision* (9) #3, pp.705–718, 2000
261. J. Pitman, *Probability*, Springer, 1993
262. J. Portilla, E. Simoncelli, "A parametric texture model based on joint statistics of complex wavelet coefficients," *Int. J. Computer Vision* (40) #1, pp.49–70, 2000
263. J. Portilla, V. Strela, M. Wainwright, E. Simoncelli, "Image denoising using scale mixtures of Gaussians in the wavelet domain," *IEEE Trans. Image Processing* (12) #11, pp.1338–1351, 2003
264. W. Press, S. Teukolsky, W. Vetterling, B. Flannery, *Numerical Recipes: The Art of Scientific Computing*, Cambridge University Press, 2007
265. L. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proc. IEEE* (77), pp.257–285, 1989
266. H. Rauch F. Tung, C. Striebel, "Maximum likelihood estimates of linear dynamic systems,"*AIAA Journal*, (3) #8, 1965
267. J. Rehg, *Visual Analysis of high DOF Articulated Objects with Application to Hand Tracking*, PhD thesis, Carnegie Mellon University, 1995
268. J. Reid, "On the method of conjugate gradients for solving linear systems," In J. Reid (Ed.), *Large Sparse Sets of Linear Equations*, Academic Press, 1971
269. D. Reid, "An algorithm for tracking multiple targets," *IEEE Trans. Automatic Control* (24) #6, pp.843–854, 1979
270. B. Ripley, *Statistical Inference for Spatial Processes*, Wiley, 1988
271. B. Ripley, *Spatial Statistics*, Wiley, 1991
272. B. Ristic, S. Arulampalam, N. Gordon, *Beyond the Kalman Filter: Particle Filters for Tracking Applications,* Artech House, 2004
273. V. Rokhlin, "Rapid solution of integral equations of classical potential theory," *J. Computational Physics* (60), pp.187–207, 1985
274. J. Romberg, H. Choi, R. Baraniuk, "Bayesian tree-structured image modeling using wavelet-domain hidden Markov models," *IEEE Trans. Image Processing* (10) #7, pp.1056–1068, 2001
275. O. Ronen, J. Rohlicek, M. Ostendorf, "Parameter estimation of dependence tree models using the EM algorithm," *IEEE Signal Processing Letters* (2) #8, 1995
276. H. Rue, L. Held, *Gaussian Markov Random Fields: Theory and Applications,* CRC Press, 2005
277. Y. Saad, M. Schultz, "GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems," *SIAM J. Sci. Stat. Comput.* #7, pp.856–869, 1986

278. Y. Saad, *Iterative methods for sparse linear systems (2nd ed.)*, SIAM, 2003
279. D. Saupe, "Algorithms for Random Fractals," in H. Peitgen, D. Saupe (Eds), *The Science of Fractal Images* Springer, 1988
280. R. Schalkoff, *Pattern recognition : statistical, structural, and neural approaches*, Wiley, 1992
281. M. Schroeder, *Fractals, Chaos, Power Laws*, Freeman, 1991
282. M. Schwartz, J. Barrett, P. Fieguth, P. Rosenkranz, M. Spina, D. Staelin, "Observations of thermal and precipitation structure in a tropical cyclone by means of passive microwave imagery near 118 GHz," *Journal of Applied Meteorology* (35) #5, pp.671–678, 1996
283. G. Shafer, *A Mathematical Theory of Evidence*, Princeton University Press, 1976
284. K. Shanmugan, A. Breipohl, *Random Signals*, Wiley, 1988
285. J. Shewchuk, "An introduction to the conjugate gradient method without the agonizing pain," (unpublished), 1994
286. E. Simoncelli, W. Freeman, E. Adelson, D. Heeger, "Shiftable multi-scale transforms," *IEEE Trans. Information Theory* (38) #2, 587–607, 1992
287. D. Simon, *Optimal state estimation: Kalman, H∞ and nonlinear approaches*, Wiley-Interscience, 2006
288. S. Sinha, B. Schunck, "A two stage algorithm for discontinuity detection," *IEEE Trans. PAMI* (14) #1, pp.36–55, 1992
289. I. Sobey, *Numerical Computation Notes* (unpublished), 2000
290. M. Srinath, P. Rajasekaran, R. Viswanathan, *An introduction to statistical signal processing with applications*, Prentice-Hall, 1996
291. J. Starck, F. Murtagh, A. Bijaoui *Image Processing and Data Analysis, The Multiscale Approach*, Cambridge University Press, 1998
292. R. Stoica, X. Descombes, J. Zerubia, "A Gibbs point process for road extraction from remotely sensed images," *Int. J. Computer Vision* (57) #2, pp.121–136, 2004
293. G. Strang, T. Nguyen, *Wavelets and Filter Banks (2nd Ed.)*, Wellesley College, 1996
294. V. Strassen, "Gaussian elimination is not optimal," *Numer. Math* (13), pp.354–356, 1969
295. C. Sutton, A. McCallum, "An introduction to conditional random fields for relational learning," *Introduction to Statistical Relational Learning*, MIT Press, 2007
296. M. Swain, D. Ballard, "Color indexing," *Int. J. Computer Vision* (7), pp.11–32, 1991
297. R. Swendsen, J. Wang, "Nonuniversal critical dynamics in Monte Carlo simulation," *Physical Review Letters* (58) #2, pp.86–88, 1987
298. R. Szeliski, *Bayesian Modeling of Uncertainty in Low-level Vision*, Kluwer Academic, 1989
299. R. Szeliski, "Fast surface interpolation using hierarchical basis functions," *IEEE Trans. PAMI* (12) #6, pp.513–528, 1990
300. R. Szeliski, D. Tonnesen, "Surface modeling with oriented particle systems," *Computer Graphics* (26) #2, pp.185–194, 1992
301. A. Tarantola, *Inverse Problem Theory and Methods for Model Parameter Estimation*, SIAM, 2004
302. D. Taubman, M. Marcellin, M. Rabbani, "JPEG2000: Image compression fundamentals, standards and practice," *J. Electronic Imaging* (11), pp.286, 2002
303. D. Terzopoulos, "Multilevel computer processes for visual surface reconstruction," *Computer Vision, Graphics, Image Processing* (24), pp.52–96, 1983
304. D. Terzopoulos, "Regularization of inverse visual problems involving discontinuities," *IEEE Trans. PAMI* (8) #4, pp.413–424, 1986
305. D. Terzopoulos, R. Szeliski, "Tracking with Kalman snakes," In *Active Vision*, MIT Press, pp.3–20, 1993

306. A. Tewfik, M. Kim, "Correlation structure of the discrete wavelet coefficients of fractional Brownian motion," *IEEE Trans. Information Theory* (38) #2, pp.904–909, 1992
307. S. Thrun, Y. Liu, D. Koller, A. Ng, Z. Ghahramani, H. Durrant-Whyte, "Simultaneous localization and mapping with sparse extended information filters," *Int. J. Robotics Research* (23) #7-8, pp.693, 2004
308. A. Tikhonov, V. Arsenin, *Solutions of Ill-Posed Problems*, Winston, 1977
309. A. Tikhonov, A. Goncharski, V. Stepanov, I. Kochikov, "Ill-posed image processing problems," *Soviet Physics - Doklady* (32), pp.456–458, 1987
310. A. Tikhonov em et al., *Numerical Methods for the Solution of Ill-Posed Problems*, Kluwer Academic Publishers, 1995
311. M. Tippett, J. Anderson, C. Bishop, T. Hamill, J. Whitaker, "Ensemble square root filters," *Monthly Weather Review* 131, pp.1485–1490, 2003
312. S. Torquato, *Random Heterogeneous Materials: Microstructure and Macroscopic Properties*, Springer, 2002
313. N. Trefethen, D. Bau, *Numerical Linear Algebra*, SIAM, 1997
314. A. Tremeau, N. Borel, "A region growing and merging algorithm to color segmentation," *Pattern Recognition* (30) #7, pp.1191–1203, 1997
315. M. Varma, A. Zisserman, "A statistical approach to material classification using image patches," *IEEE Trans. PAMI* (31) #11, pp.2032–2047, 2009
316. O. Vasilyev, N. Kevlahan, "An adaptive multilevel wavelet collocation method for elliptic problems," *J. Computational Physics* (206) #2, pp.412–431, 2005
317. B. Vidakovic, *Statistical Modeling by Wavelets*, Wiley, 1999
318. C. Vieren, F. Cabestaing, J. Postaire, "Catching moving objects with snakes for motion tracking," *Pattern Recognition Letters* (16) #7, pp.679–685, 1995
319. P. Viola, W. Wells, "Alignment by maximization of mutual information," *Int. J. Computer Vision* (24) #2, pp.137–154, 1997
320. A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Information Theory* (13) #2, pp.260–269, 1967
321. H. van der Vorst, *Iterative Krylov Methods for Large Linear Systems*, Cambridge University Press, 2003
322. G. Wahba, "Practical approximate solutions to linear operator equations when the data are noisy," *SIAM J. Numerical Analysis* (14), 1977
323. G. Wahba, "Bayesian "Confidence Intervals" for the Cross-Validated Smoothing Spline," *J. Royal Statistical Society B* (45), pp.133–150, 1983
324. G. Wahba, *Spline Models for Observational Data*, SIAM Series in Applied Mathematics #59, SIAM, 1990
325. E. Wan, R. van der Merwe, A. Nelson, "Dual estimation and the unscented transformation," in *Advances in Neural Information Processing Systems 12* (S. Solla, T. Leen, K. Mller, Eds.), MIT Press, pp.666–672, 2000
326. E. Wan, R. van der Merwe, "The unscented Kalman filter," in *Kalman filtering and Neural Networks*, Wiley, pp.221–280, 2001
327. Y. Wang, Q. Ji, "A dynamic conditional random field model for object segmentation in image sequences," *CVPR*, 2005
328. E. Wegman, D. DePrest, *Statistical Image Processing and Graphics* Marcel Dekker, 1986
329. S. Wesolkowski, P. Fieguth, "Hierarchical regions for image segmentation," *ICIAR*, Portugal, 2004
330. P. Wesseling, *An Introduction to Multigrid Methods*, Wiley, 1991
331. G. Whitten, "Scale space tracking and deformable sheet models for computational vision," *IEEE Trans. PAMI* (15) #7, 1993

332. K. Wilson, "Problems in physics with many scales of length," *Scientific American* (241), pp.158–179, 1979
333. A. Willsky, *6.433 – Recursive Estimation*, *Department of Electrical Engineering & Computer Science*, Massachusetts Institute of Technology, 1992
334. A. Willsky, "Multiresolution Markov models for signal and image processing," *Proc. IEEE* (90) #8, pp.1396–1458, 2002
335. G. Winkler, *Image Analysis, Random Fields, and Dynamic Monte Carlo Methods (2nd ed.)*, Springer, 2003
336. C. Won, R. Gray, *Stochastic Image Processing*, Springer, 2004
337. J. Woods, C. Radewan, "Kalman filtering in two dimensions," *IEEE Trans. Information Theory* (23), pp.437–481, 1977
338. J. Woods, V. Ingle, "Kalman filtering in two dimensions: Further results," *IEEE Trans. Acoustics, Speech, Signal Processing* (29), pp.188–197, 1981
339. G. Wornell, A. Oppenheim, "Estimation of fractal signals from noisy measurements using wavelets," *IEEE Trans. Signal Processing* (40), pp.611–623, 1992
340. G. Wornell, "Wavelet-based representation for the $1/f$ family of fractal processes," *Proc. IEEE*, 1993
341. C. Wu, Peter C. Doerschuk, "Tree approximations to markov random fields," *IEEE Trans. PAMI* (17) #4, pp.391–402, April 1995
342. C. Wunsch, E. Gaposchkin, "On using satellite altimetry to determine the general circulation of the oceans with application to geoid improvement," *Reviews of Geophysics and Space Physics* (18) #4, pp.725–745, 1980
343. C. Wunsch, "Sampling characteristics of satellite orbits," *J. Atmospheric and Oceanic Tech.* (6)#6, pp.891–907, 1989
344. R. Yager, M. Fedrizzi, J. Kacprzyk (Eds.), *Advances in the Dempster-Shafer Theory of Evidence*, Wiley, 1994
345. M. Yaou, W. Chang, "Fast surface interpolation using multiresolution wavelet transform," *IEEE Trans. PAMI* (16) #7, pp.673–688, 1994
346. D. Young, *Iterative Solution of Large Linear Systems*, Academic Press, 1971
347. H. Yserentant, "On the multi-level splitting of finite element spaces," *Numerische Mathematik* (49) #4, pp.379–412, 1986
348. H. Yserentant, "Two preconditioners based on the multilevel splitting of finite element spaces," *Numerische Mathematik* (58) #2, pp.163–184, 1990
349. Q. Zheng, R. Chellappa, "Estimation of illuminant direction, albedo, and shape from shading," *IEEE Trans. PAMI* (13) #7, pp.680–702, 1991

Index

ABCD lemma 68, 99, 388

Banded matrices *see* Sparsity
Basis
  Change *see* Change of basis
  Definition 384
  Reduction *see* Dimensionality reduction
Bayesian problems
  Approximate 70
  Definition 31
  Dynamic 42, 86
  Estimation 64
  Least squares 45, 65
  Linear least squares 45, 66
  MAP 45
  Regularization 37
  Static 40, 67
Boundary conditions 153

Canonical problems
  Data fusion 41
  Dynamic 42
  Static 40
Change of basis 241
  Explicit 244
  Hierarchical 269, 276
  Implicit 245
  Kernels 254
  Orthogonal 245
Cholesky decomposition 295, 401, 407
Circulance 262, 392
  Circular convolution 427
Condition number 27, 401
Conditional random fields 225
Conditioning 23, 25, 27, 206, 241, 310
Conjugate gradient 306
Convolution 145, 424, 427
  Circular 427
Coupling 135, 242, 281
Covariance matrices 37, 70, 389, 419, 420
  Approximated 167
  Definition 414
  Diagonal 418
  Eigendecomposition 399
  Inverse and Markovianity 188
  Positive-definite 389, 390
Cross validation 37, 38, 83

Data fusion 16, 41, 76, 83, 377
Deterministic problems *see* Non-Bayesian problems
Dimensionality reduction 135, 247, 250, 253
Duality
  Bayesian–non-Bayesian 39, 73
  Prior–measurement 73
  Static–dynamic 89, 326
Dynamic problems
  Canonical 42, 86
  Estimation 85, 97, 100
  Sampling 118, 357

Eigendecompositions 248, 262, 304, 305, 396, 420
Estimation 46, 69
  Approximate 70
  Bayesian 40, 45, 64
  Definition 44

Discrete state 119
Dynamic 42, 85, 97, 100
Non-Bayesian 48, 58
Static 40, 57
Using Fourier transform 266
Expectation 64, 186, 411
Expectation-Maximization (EM) 232

Forward problems 14, 78
Fourier transform 262, 361, 428

Gauss–Markov processes 88, 181
Gauss–Markov random fields 185, 189
Gaussian 419
Multivariate distribution 417
Univariate distribution 412
Gaussian elimination 295, 402
Gaussian transformation 402
Gibbs random fields 192
Comparison to Markov 194
Ising 228, 283
Sampling 366
Gram–Schmidt 309, 404
Graphical coupling 135, 242, 281

Hidden Markov models 215
Modelling 231
Hierarchical methods
Change of basis 269, 276
Discrete state fields 281, 373
Interpolated bases 272
Markov random fields 278
MCMC 370
Multigrid 313
Multiscale 339, 363
Nested dissection 296
Overview 137
Wavelets 273

Ill-Conditioned problems *see* Conditioning
Ill-Posed problems *see* Posedness, Inverse problems
Image processing 22, 423
Blurring 21, 24, 425
Convolution 424, 427
Denoising 29, 216, 347, 431
Edge detection 221, 425, 431
Fourier transform 428
Segmentation 219, 233, 432
Wavelet transform 273, 428
Interpolation 9
Cross validation 38
Dynamic 98, 112
Iterative 302
Multidimensional 159
Posedness 20
Regularization 32
Inverse problems 2, 13
Bayesian *see* Bayesian problems
Canonical *see* Canonical problems
Conditioning 23, 25
Dynamic 42
Non-Bayesian *see* Non-Bayesian problems
Posedness 19, 30
Regularization 29
Static 40
Iterative solvers *see* Linear systems methods

Kalman filter 97, 98, 100, 325, 330
Derivation 93
Information form 102
Marching 327, 332
Nonlinear 114
Reduced order 338
Reduced update 336
Smoother 109, 112, 331
Square root 103
Steady state 107, 334
Stripped 334
Kernels *see* Sparsity
Kriging 45, 164

Least squares 30, 48, 62, 409
Bayesian 45, 65
Linear 58
Pseudoinverse 61
Regularized 61
Weighted 48, 61
Lexicographic ordering 133
Linear dependence 19, 384
Linear regression 23, 62
Linear systems 245, 293
Overdetermined 22, 409
Underdetermined 22, 409
Linear systems methods 293
Cholesky decomposition 295, 401

Conjugate gradient 306
Gauss–Jacobi 299
Gauss–Seidel 299, 302
Gaussian elimination 295, 402
LU decomposition 403
Multigrid 313
Nested dissection 296
Preconditioning 310
Successive overrelaxation (SOR) 303
LU decomposition 403

Marching methods 327, 362
Markov chains 119, 181
Markovianity 179, 222, 297, 327
Gauss–Markov process 88, 181
Gauss–Markov random field 185
Hidden 215, 223
Markov random field 182
One-dimensional 180
Sparsity 188
Matlab XV, 10
Matrix
Block-matrix identities 387
Circulance *see* Circulance
Condition number 27, 401
Conditioning 23, 27, 310
Covariance *see* Covariance
Derivatives 394, 395
Identities 387
Inequality 389
Jacobian 394
Kernels *see* Sparsity
List of types 392, 393
Null space 19, 384
Positivity *see* Positive-definiteness
Pseudoinverse 409
Range space 19, 385
Rank 19, 385
Sparse *see* Sparsity
Square root 166, 401, 407
Matrix transformations 395
Cholesky decomposition 295, 401
Eigendecomposition 396
Gaussian 402
Gaussian elimination 295, 402
Gram–Schmidt 404
LU decomposition 403
QR 104, 406
Square root 407
Maximum a Posteriori (MAP) 45
Maximum likelihood (ML) 48, 63
MCMC *see* Monte Carlo
Membrane prior 35, 150
Model inference 169, 170, 199, 206
Parameter estimation 50
Modelling 148
Bayesian 158
Boundary conditions 153
Dynamic 166, 325
Hidden fields 231
Markov random field 199, 204
Multidimensional 134
Non-Bayesian 149
Nonstationary 164
Representation 172, 207
Monte Carlo 355, 365
Multigrid 313
Multiscale 339, 363

Non-Bayesian problems
Definition 31
Estimation 58
Maximum likelihood 63
Modelling 149
Regularization 34, 61
Normal distribution *see* Gaussian
Null space 19, 384

Orthogonality 59, 65

Parameter estimation *see* Model inference
Posedness 23, 30
Positive-definiteness 388
Analytical forms 160
Definition 392
Intuition 27, 390
Posterior sampling *see* Sampling
Preconditioning 310
Principal components 248, 252
Prior
Boundaries 153
Membrane 35, 150
Model, Bayesian 37, 158
Model, non-Bayesian 34, 150
Sampling *see* Sampling
Thin-plate 35, 150
Zero mean 40, 68, 71
Prior-Free problems *see* Non-Bayesian problems

Pseudoinverses 61, 247, 409

QR decomposition 104, 406

Random fields 6
  Causal 189
  Conditional 225
  Definition 415
  Discrete state 227
  Gibbs 192
  Markov 182
  Markov vs. Gibbs 194
  Modelling 199
  Noncausal 185
  Stationary 415
Random variables 411
  Correlation 413
  Independence 412
Random vectors 414
  Covariance 414
  Transformation 416
Range space 19, 385
Rank
  Full column 19, 385, 409
  Full row 19, 385
Regularization 29
  Bayesian 37
  Non-Bayesian 34
  Tikhonov 30

Sampling 355, 362, 374, 408
  Discrete state 366, 370
  Dynamic 118, 357
  Examples 46, 75, 92
  Posterior 49, 74
  Prior 42, 74, 374
  Static 74, 358
  Using Fourier transform 266
Simulated annealing 366, 369, 370, 372
Singular value decomposition 400
Singular value decompositions 27
Sparsity
  Banded matrices 139, 144
  Computational complexity 144
  Markovianity 188
  Matrix kernels 141, 146, 151
  Sparse matrices 139
Square root matrices 103, 401, 407
Static problems
  Bayesian 67
  Canonical 40
  Estimation 57
  Non-Bayesian 58
  Sampling 74, 358
Stationarity 141, 160, 164, 415
SVD *see* Singular value decomposition

Thin-plate prior 35, 150
Tikhonov regularization 30
Tomography 50

Validation 35
Variogram 45
Viterbi 120

Wavelet transform 273, 347, 428
  Wavelets and statistics 275