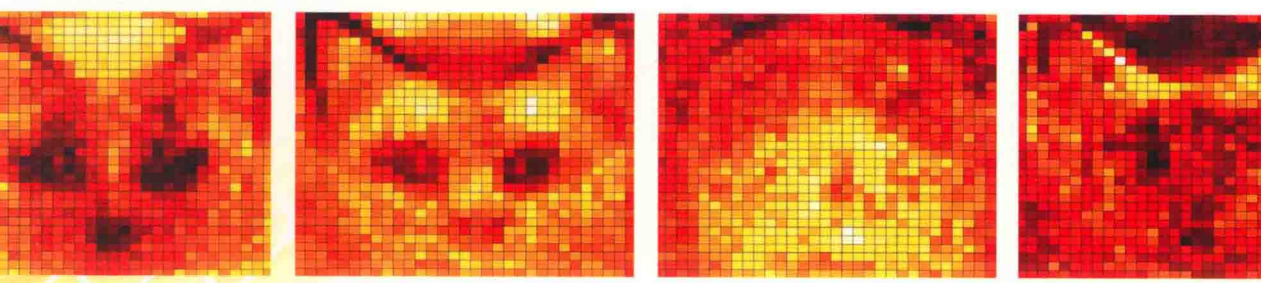
OXFORD

Data-Driven Modeling & Scientific Computation

Methods for Complex Systems & Big Data



J. NATHAN KUTZ

Data-Driven Modeling & Scientific Computation

Methods for Complex Systems & Big Data

J. NATHAN KUTZ

Department of Applied Mathematics
University of Washington

常州大学山书馆
藏书章

OXFORD
UNIVERSITY PRESS

OXFORD
UNIVERSITY PRESS

Great Clarendon Street, Oxford, OX2 6DP,
United Kingdom

Oxford University Press is a department of the University of Oxford.
It furthers the University's objective of excellence in research, scholarship,
and education by publishing worldwide. Oxford is a registered trade mark of
Oxford University Press in the UK and in certain other countries

© J. Nathan Kutz 2013

The moral rights of the author have been asserted

First Edition published in 2013

Impression: 1

All rights reserved. No part of this publication may be reproduced, stored in
a retrieval system, or transmitted, in any form or by any means, without the
prior permission in writing of Oxford University Press, or as expressly permitted
by law, by licence or under terms agreed with the appropriate reprographics
rights organization. Enquiries concerning reproduction outside the scope of the
above should be sent to the Rights Department, Oxford University Press, at the
address above

You must not circulate this work in any other form
and you must impose this same condition on any acquirer

Published in the United States of America by Oxford University Press
198 Madison Avenue, New York, NY 10016, United States of America

British Library Cataloguing in Publication Data
Data available

Library of Congress Control Number: 2013937977

ISBN 978–0–19–966033–9 (hbk.)
ISBN 978–0–19–966034–6 (pbk.)

Printed and bound by
CPI Group (UK) Ltd, Croydon, CR0 4YY

Links to third party websites are provided by Oxford in good faith and
for information only. Oxford disclaims any responsibility for the materials
contained in any third party website referenced in this work.

Dedication

For Kristy, Lauren and Katie

Acknowledgments

The idea of the first part of this book began as a series of conversations with Dave Muraki. It then grew into the primary set of notes for a scientific computing course whose ambition was to provide a truly versatile and useful course for students in the engineering, biological and physical sciences. And over the last couple of years, the book expanded to included methods for data analysis, thus bolstering the intellectual scope of the book significantly. Unbeknownst to them, much of the data analysis portion of the book was heavily inspired by the fantastic works of Emmanuel Candés, Yannis Kevrekidis and Clancy Rowley and various conversations I had with each of them. I've also benefitted greatly from early discussions with James Rossmanith, and with implementation ideas with Peter Blossey and Sorin Mitran; and more recently on dimensionality reduction methods with Steven Brunton, Edwin Ding, Joshua Proctor, Peter Schmid, Eli Shlizerman, Jonathan Tu and Matthew Williams. Leslie Butson, Sarah Hewitt and Jennifer O'Neil have been very helpful in editing the book so that it is more readable, useful and error-free. A special thanks should also be given to all the many wonderful students who have provided so much critical commentary and vital feedback for improving the delivery, style and correctness of the book. Of course, all errors in this book are the fault of my daughters' hamsters Fluffy and Quickles.

试读结束：需要全本请在线购买： www.ertongbook.com

Prolegomenon

Scientific computing is ubiquitous in the physical, biological and engineering sciences. Today, proficiency with computational methods, or lack thereof, can have a major impact on a researcher's ability to effectively analyze a given problem. Although a host of numerical analysis courses are traditionally offered in the mathematical sciences, the typical audience is the professional mathematician. Thus the emphasis is on establishing proven techniques and working through rigorous stability arguments, for instance. No doubt, this vision of numerical analysis is essential and provides the basic groundwork for most numerical analysis courses and books. This more traditional approach to the teaching of numerical methods generally requires more than a year in coursework to achieve a level of proficiency necessary for solving practical problems since the focus is on establishing rigor versus implementation of the methods with a high-level programming language.

The goal of this book is to embark on a new tradition: establishing computing proficiency as the first and foremost priority above rigorous analysis. Thus the major computational methods established over the past few decades are considered with emphasis on their use and implementation versus their rigorous analytic framework. A terse time-frame is also necessary in order to effectively augment the education of students from a wide variety of scientific departments. The three major techniques for solving partial differential equations are all considered: finite differences, finite elements and spectral methods. And the addition in this manuscript of data analysis techniques represents a fairly radical departure from the standard curriculum in mathematics and applied mathematics departments.

MATLAB has established itself as the leader in scientific computing software. The built-in algorithms developed by MATLAB allow the computational focus to shift from technical details to overall implementation and solution techniques. Heavy and repeated use is made of MATLAB's linear algebra packages, fast Fourier transform routines, and finite element (partial differential equations) package. These routines are the workhorses for most solution techniques and are treated to a large extent as blackbox operations. Of course, cursory explanations are given of the underlying principles in any of the routines utilized, but it is largely left as reference material in order to focus on the application of the routine. It is assumed that when necessary, one could implement these techniques by using standard libraries from, for instance, LAPACK.

The end goal is for the student to develop a sense of confidence about implementing computational techniques. Specifically, at the end of the book, the student should be able to solve almost any 1D, 2D or 3D problem of the elliptic, hyperbolic or parabolic type. Or at the least, they should have a great deal of knowledge about how to solve the problem and should have enough information and references at their disposal to circumvent any implementation difficulties. Likewise, with the data analysis framework presented, the key concepts of statistics, time–frequency analysis and low dimensional reduction (SVD) should allow one to have an excellent starting point for moving forward in a given problem where data plays a critical role.

Overall, the combination of methods for solving complex spatio-temporal systems along with understanding how to integrate data into the analysis provides an excellent framework for

integrating dynamics of complex systems with big data. This is especially true in the later chapters (Chapters 18–23) where a clear integration of data methods and dynamics is advocated. Given the growing importance of big data methods, it is essential that scientific computing keep pace with some of the exciting developments in this field, especially as it relates to more traditional scientific computing applications.

How to Use This Book

There are a number of intended audiences for this book as is shown in the division of the book into three numerical methods parts and one applications portion. Indeed, the various parts of the book were developed with different student audiences in mind. In what follows, the specific target audiences are highlighted along with the portions of the book appropriate for their use. Ultimately, this gives a roadmap on how to use this book for one's desired ends. All portions of the book have been heavily vetted by undergraduate and graduate students alike, thus improving the overall readability and usefulness. What is unique about this book is the applications portion which attempts to allow students to solve real problems of broad scientific interest. My own personal frustration with the myriad of MATLAB computing books is that many of the problems designed or used for illustration of key concepts are fairly uninspiring and devoid of real-world applicability. More will be said about this in the following paragraphs.

Undergraduate course in beginning scientific computing

The first part of this book reflects the topical coverage of material taught at the University of Washington campus for freshman–sophomore level engineering and physical science students. Given that MATLAB has become the programming language of choice in our engineering programs, the coverage begins by considering basic concepts and theoretical ideas in the context of programming language infrastructure. Thus programming and algorithm development becomes an integral part of developing practical routines for computing, for instance, least-square fits, derivatives or integrals. Moreover, simple things like making nice plots, generating movies and importing/exporting data files with MATLAB are incorporated into the pedagogical structure. Thus Chapters 1 through 6 give a basic introduction to MATLAB, to programming infrastructure (**if** and **for** loops) and to problem solving development. In addition to these first five chapters, Chapter 6 on differential equations is covered towards the end of the book. The ability to quickly and efficiently solve differential equations is critical for many junior and senior level courses in the engineering sciences such as aeronautics/astronautics (the three-body problem) or electrical engineering (circuit theory). Thus the first part of this book has a well-defined, beginning scientific computing audience.

Graduate course in scientific computing

In addition to beginning students, there is a great deal of effort in providing an educational infrastructure and high-level overview of scientific computing methods to graduate students (or advanced undergraduates) from a broad range of scientific disciplines. A traditional graduate course in numerical analysis, while extremely relevant, often is focused on the mathematical infrastructure versus practical implementation. Further, many numerical analysis sequences are quite devoid of engineering, biological and physical science students. The second part of this book, Chapters 7 through 11, provides a high-level introduction to the computational methods used for solving differential and partial differential equations. It certainly is the case that such

systems provide an underlying theoretical framework for many applications of broad scientific interest. Thus the key elements of finite difference, spectral and finite elements are all considered. The starting point for the graduate students is in Chapter 7 with stepping techniques for differential equations. On the one hand, one can envision skipping all of the first part of the book to begin the graduate level treatment of scientific computing. But on the other hand, the inclusion of Part I of this book allows graduate students to have a refresher section for simple things such as plotting, constructing movies, importing/exporting data, or simply reviewing programming architecture. Thus the graduate student can use the first part of this book as a reference for their intended studies.

Graduate course in computational methods for data analysis

Parts I and II of this book offer a fairly standard treatment, although slanted heavily towards implementation here, of beginning and advanced numerical methods leading to the construction of numerical solutions of partial differential equations. In contrast, Part III of this book (Chapters 12 through 23) offers a unique perspective on data analysis methods. Indeed, one would be hard pressed to find such a treatment in any current textbook in the mathematical sciences. The aim of this third part is to introduce graduate students (or advanced undergraduates) in the sciences to the burgeoning field of data analysis. This area of research is expanding at an incredible pace in the sciences due to the proliferation of data collection in almost every field of science. The enormous data sets routinely encountered in the sciences now certainly give enormous incentive to their synthesis, interpretation and conjectured meaning. This portion of the book attempts to bring together in a self-consistent fashion the key ideas from (i) statistics, (ii) time–frequency analysis and (iii) low dimensional reductions in order to provide meaningful insight into the data sets one is faced with in any scientific field today. This is a tremendously exciting area and much of this part of the book is driven by intuitive examples of how the three areas (i)–(iii) can be used in combination to give critical insight into the fundamental workings of various problems. As with Part II of this book, access to the introductory material in Part I allows students from various backgrounds to supplement their background where appropriate, thus making Part I an indispensable part of the overall architecture of the book.

Computational methods reference guide

In addition to its primary use as a textbook for either a graduate or undergraduate course in scientific computing or data analysis, the book also serves as a helpful reference guide. As a reference guide, its strength lies in either giving the appropriate high-level overview necessary along with key pieces of MATLAB code for implementation, or the scientific applications portion of the book provides example ideas and techniques for solving a broad class of problems. Using either the applications or theory sections, or both in combination, provides an effective and quick way to refresh one's skills and/or analytic understanding of a solutions technique. In practice, the most common comment I hear concerning this book is that students have found them useful long after their course work has been completed. I believe this is attributed to the low entry threshold for obtaining both practical theoretical knowledge and key snippets of MATLAB code for use in developing a piece of code beyond what is covered in the text. Further, since many high-level

MATLAB subroutines are introduced, a person referencing the book can be assured of exposure to techniques well beyond the simple and trivial methods that might be at first considered.

Scientific applications—Bringing it all together

The most critical aspect of this book is the scientific applications part (Chapters 24 through 26). The philosophy here is simple: solve real problems. There is something a bit disingenuous to me about developing sophisticated methodology and then applying it to either (i) highly contrived examples, or (ii) greatly oversimplified problems that are constructed for theoretical/analytical convenience. It is well understood that each of these serve their purpose in pedagogy. However, if the aim is to make one proficient in building real code, then real problems, with all their complexities, must be considered. The selection of problems is broad enough that an instructor should be able to pick something of interest to themselves. Further, each problem has a well laid out background so that the problem is developed in its appropriate context. The level of difficulty of each problem increases as the problem is developed so that the same problem can serve for undergraduates, advanced undergraduates or graduate students alike, the only difference being in where the student was asked to stop. This allows students the ability to push beyond the basic formulation if they desire. Moreover, there is a context in which their computations are placed, making connection to important historical problems such as the three-body problem, quantum mechanics, 2D advection–diffusion, etc. It is hoped that such problem formulation is much more exciting for both students and faculty alike. This is a unique way in which to think about implementing the theoretical and computational tools learned in the book.

About MATLAB

MATLAB (**MAT**rix **LAB**oratory) has become the *tool of choice* for the teaching of and rapid prototyping of a myriad of problems arising in the physical, engineering and biological sciences. As a high-level language rooted in matrix and vector mathematics, it provides an exceptional integrated programming environment for algorithm development, data analysis, visualization and numerical computation. Using MATLAB, you can solve technical computing problems at a fraction of the programming effort required with traditional languages such as C, C++ or Fortran.

The objective of this book is to provide methods for computationally solving problems, and then to actually solve them. As such, a high-level scientific language is ideal for generating small, yet extremely powerful algorithms. Moreover, full advantage is taken of professionally developed algorithms that few students could match in terms of efficiency, accuracy and/or cutting-edge relevance. Armed with such high-level program tools, rapid and significant progress can be made in solving significant problems that arise in the sciences. Thus the focus is on solving the scientific problem versus the algorithm implementation and its nuances in accuracy, stability and speed.

MATLAB is not free. It is developed by Mathworks Inc. based outside of Boston, MA. It is a privately held company that today has over one million users from all walks of life and industries. A student version is available online and/or from select bookstores, and is highly recommended, for those studying at any institution of learning (www.mathworks.com).

Given its dominance and importance, it is only natural that there should be a MATLAB-styled version through open source development. GNU Octave is a high-level interpreted language, primarily intended for numerical computations. Much like MATLAB, it provides capabilities for the numerical solution of linear and nonlinear problems, and for performing other numerical experiments. It also provides extensive graphics capabilities for data visualization and manipulation. Octave is normally used through its interactive command line interface, but it can also be used to write noninteractive programs. The Octave language is quite similar to MATLAB so that most programs are easily portable. Octave is free, although there is a suggested donation.

Whether it be MATLAB or Octave, a high-level interpreted language is assumed to be available for your use in going through this book. The availability of such high-level languages is transformative in one's ability to rapidly implement and develop the sophisticated computing algorithms developed herein. It is my view that such high-level languages let one much more clearly see the forest through the trees.

试读结束：需要全本请在线购买： www.ertongbook.com

Contents

Prolegomenon xiii
How to Use This Book xv
About MATLAB xviii

PART I Basic Computations and Visualization

1 MATLAB Introduction **3**

1.1 Vectors and Matrices 3
1.2 Logic, Loops and Iterations 9
1.3 Iteration: The Newton–Raphson Method 13
1.4 Function Calls, Input/Output Interactions and Debugging 18
1.5 Plotting and Importing/Exporting Data 23

2 Linear Systems **31**

2.1 Direct Solution Methods for $Ax = b$ 31
2.2 Iterative Solution Methods for $Ax = b$ 35
2.3 Gradient (Steepest) Descent for $Ax = b$ 39
2.4 Eigenvalues, Eigenvectors and Solvability 44
2.5 Eigenvalues and Eigenvectors for Face Recognition 49
2.6 Nonlinear Systems 56

3 Curve Fitting **61**

3.1 Least-Square Fitting Methods 61
3.2 Polynomial Fits and Splines 65
3.3 Data Fitting with MATLAB 69

4 Numerical Differentiation and Integration **77**

4.1 Numerical Differentiation 77
4.2 Numerical Integration 83
4.3 Implementation of Differentiation and Integration 87

5 Basic Optimization 93

5.1 Unconstrained Optimization (Derivative-Free Methods) 93
5.2 Unconstrained Optimization (Derivative Methods) 99
5.3 Linear Programming 105
5.4 Simplex Method 110
5.5 Genetic Algorithms 113

6 Visualization 119

6.1 Customizing Plots and Basic 2D Plotting 119
6.2 More 2D and 3D Plotting 125
6.3 Movies and Animations 131

PART II Differential and Partial Differential Equations

7 Initial and Boundary Value Problems of Differential Equations 137

7.1 Initial Value Problems: Euler, Runge–Kutta and Adams Methods 137
7.2 Error Analysis for Time-Stepping Routines 144
7.3 Advanced Time-Stepping Algorithms 149
7.4 Boundary Value Problems: The Shooting Method 153
7.5 Implementation of Shooting and Convergence Studies 160
7.6 Boundary Value Problems: Direct Solve and Relaxation 164
7.7 Implementing MATLAB for Boundary Value Problems 167
7.8 Linear Operators and Computing Spectra 172

8 Finite Difference Methods 180

8.1 Finite Difference Discretization 180
8.2 Advanced Iterative Solution Methods for $Ax = b$ 186
8.3 Fast Poisson Solvers: The Fourier Transform 186
8.4 Comparison of Solution Techniques for $Ax = b$: Rules of Thumb 190
8.5 Overcoming Computational Difficulties 195

9 Time and Space Stepping Schemes: Method of Lines 200

9.1 Basic Time-Stepping Schemes 200
9.2 Time-Stepping Schemes: Explicit and Implicit Methods 205
9.3 Stability Analysis 209

9.4 Comparison of Time-Stepping Schemes 213
9.5 Operator Splitting Techniques 216
9.6 Optimizing Computational Performance: Rules of Thumb 219

10 Spectral Methods 225

10.1 Fast Fourier Transforms and Cosine/Sine Transform 225
10.2 Chebychev Polynomials and Transform 229
10.3 Spectral Method Implementation 233
10.4 Pseudo-Spectral Techniques with Filtering 235
10.5 Boundary Conditions and the Chebychev Transform 240
10.6 Implementing the Chebychev Transform 244
10.7 Computing Spectra: The Floquet–Fourier–Hill Method 249

11 Finite Element Methods 256

11.1 Finite Element Basis 256
11.2 Discretizing with Finite Elements and Boundaries 261
11.3 MATLAB for Partial Differential Equations 266
11.4 MATLAB Partial Differential Equations Toolbox 271

PART III Computational Methods for Data Analysis

12 Statistical Methods and Their Applications 279

12.1 Basic Probability Concepts 279
12.2 Random Variables and Statistical Concepts 286
12.3 Hypothesis Testing and Statistical Significance 294

13 Time–Frequency Analysis: Fourier Transforms and Wavelets 301

13.1 Basics of Fourier Series and the Fourier Transform 301
13.2 FFT Application: Radar Detection and Filtering 308
13.3 FFT Application: Radar Detection and Averaging 316
13.4 Time–Frequency Analysis: Windowed Fourier Transforms 322
13.5 Time–Frequency Analysis and Wavelets 328
13.6 Multi-Resolution Analysis and the Wavelet Basis 335
13.7 Spectrograms and the Gábor Transform in MATLAB 340
13.8 MATLAB Filter Design and Wavelet Toolboxes 346

14 Image Processing and Analysis 358

14.1 Basic Concepts and Analysis of Images 358
14.2 Linear Filtering for Image Denoising 364
14.3 Diffusion and Image Processing 369

15 Linear Algebra and Singular Value Decomposition 376

15.1 Basics of the Singular Value Decomposition (SVD) 376
15.2 The SVD in Broader Context 381
15.3 Introduction to Principal Component Analysis (PCA) 387
15.4 Principal Components, Diagonalization and SVD 391
15.5 Principal Components and Proper Orthogonal Modes 395
15.6 Robust PCA 403

16 Independent Component Analysis 412

16.1 The Concept of Independent Components 412
16.2 Image Separation Problem 419
16.3 Image Separation and MATLAB 424

17 Image Recognition: Basics of Machine Learning 431

17.1 Recognizing Dogs and Cats 431
17.2 The SVD and Linear Discrimination Analysis 436
17.3 Implementing Cat/Dog Recognition in MATLAB 445

18 Basics of Compressed Sensing 449

18.1 Beyond Least-Square Fitting: The L^1 Norm 449
18.2 Signal Reconstruction and Circumventing Nyquist 456
18.3 Data (Image) Reconstruction from Sparse Sampling 464

19 Dimensionality Reduction for Partial Differential Equations 472

19.1 Modal Expansion Techniques for PDEs 472
19.2 PDE Dynamics in the Right (Best) Basis 478
19.3 Global Normal Forms of Bifurcation Structures in PDEs 482
19.4 The POD Method and Symmetries/Invariances 492
19.5 POD Using Robust PCA 499

20 Dynamic Mode Decomposition 506

20.1 Theory of Dynamic Mode Decomposition (DMD) 506
20.2 Dynamics of DMD Versus POD 510
20.3 Applications of DMD 515

21 Data Assimilation Methods 521

21.1 Theory of Data Assimilation 521
21.2 Data Assimilation, Sampling and Kalman Filtering 526
21.3 Data Assimilation for the Lorenz Equation 529

22 Equation-Free Modeling 537

22.1 Multi-Scale Physics: An Equation-Free Approach 537
22.2 Lifting and Restricting in Equation-Free Computing 542
22.3 Equation-Free Space–Time Dynamics 547

23 Complex Dynamical Systems: Combining Dimensionality Reduction, Compressive Sensing and Machine Learning 551

23.1 Combining Data Methods for Complex Systems 551
23.2 Implementing a Dynamical Systems Library 556
23.3 Flow Around a Cylinder: A Prototypical Example 564

PART IV Scientific Applications

24 Applications of Differential Equations and Boundary Value Problems 573

24.1 Neuroscience and the Hodgkin–Huxley Model 573
24.2 Celestial Mechanics and the Three-Body Problem 577
24.3 Atmospheric Motion and the Lorenz Equations 581
24.4 Quantum Mechanics 585
24.5 Electromagnetic Waveguides 588

25 Applications of Partial Differential Equations 590

25.1 The Wave Equation 590
25.2 Mode-Locked Lasers 593
25.3 Bose–Einstein Condensates 600

25.4 Advection–Diffusion and Atmospheric Dynamics 604
25.5 Introduction to Reaction–Diffusion Systems 611
25.6 Steady State Flow Over an Airfoil 616

26 Applications of Data Analysis 620

26.1 Analyzing Music Scores and the Gábor Transform 620
26.2 Image Denoising through Filtering and Diffusion 622
26.3 Oscillating Mass and Dimensionality Reduction 625
26.4 Music Genre Identification 626

References 629
Index of MATLAB Commands 634
Index 636

试读结束：需要全本请在线购买： www.ertongbook.com