

Numerical Recipes in C

Second Edition

Numerical Recipes in C

The Art of Scientific Computing

Second Edition

William H. Press

Harvard-Smithsonian Center for Astrophysics

Saul A. Teukolsky

Department of Physics, Cornell University

William T. Vetterling

Polaroid Corporation

Brian P. Flannery

EXXON Research and Engineering Company

CAMBRIDGE UNIVERSITY PRESS

Cambridge New York Port Chester Melbourne Sydney

Published by the Press Syndicate of the University of Cambridge
The Pitt Building, Trumpington Street, Cambridge CB2 1RP
40 West 20th Street, New York, NY 10011-4211, USA
477 Williamstown Road, Port Melbourne, VIC, 3207, Australia

Copyright © Cambridge University Press 1988, 1992
except for §13.10 and Appendix B, which are placed into the public domain,
and except for all other computer programs and procedures, which are
Copyright © Numerical Recipes Software 1987, 1988, 1992, 1997, 2002
All Rights Reserved.

Some sections of this book were originally published, in different form, in *Computers in Physics* magazine, Copyright © American Institute of Physics, 1988–1992.

First Edition originally published 1988; Second Edition originally published 1992.
Reprinted with corrections, 1993, 1994, 1995, 1997, 2002.
This reprinting is corrected to software version 2.10

Printed in the United States of America
Typeset in \TeX

Without an additional license to use the contained software, this book is intended as a text and reference book, for reading purposes only. A free license for limited use of the software by the individual owner of a copy of this book who personally types one or more routines into a single computer is granted under terms described on p. xvii. See the section “License Information” (pp. xvi–xviii) for information on obtaining more general licenses at low cost.

Machine-readable media containing the software in this book, with included licenses for use on a single screen, are available from Cambridge University Press. See the order form at the back of the book, email to “orders@cup.org” (North America) or “directcustserv@cambridge.org” (rest of world), or write to Cambridge University Press, 110 Midland Avenue, Port Chester, NY 10573 (USA), for further information.

The software may also be downloaded, with immediate purchase of a license also possible, from the Numerical Recipes Software Web Site (<http://www.nr.com>). Unlicensed transfer of Numerical Recipes programs to any other format, or to any computer except one that is specifically licensed, is strictly prohibited. Technical questions, corrections, and requests for information should be addressed to Numerical Recipes Software, P.O. Box 380243, Cambridge, MA 02238-0243 (USA), email “info@nr.com”, or fax 781 863-1739.

Library of Congress Cataloging in Publication Data

Numerical recipes in C : the art of scientific computing / William H. Press

... [et al.]. – 2nd ed.

Includes bibliographical references (p.) and index.

ISBN 0-521-43108-5

1. Numerical analysis–Computer programs. 2. Science–Mathematics–Computer programs.

3. C (Computer program language) I. Press, William H.

QA297.N866 1992

519.4'0285'53–dc20

92-8876

A catalog record for this book is available from the British Library.

ISBN 0 521 43108 5 Book
ISBN 0 521 43720 2 Example book in C
ISBN 0 521 75037 7 C/C++ CDROM (Windows/Macintosh)
ISBN 0 521 75035 0 Complete CDROM (Windows/Macintosh)
ISBN 0 521 75036 9 Complete CDROM (UNIX/Linux)

Contents

| | |
|---|-------------------|
| <i>Preface to the Second Edition</i> | <i>xi</i> |
| <i>Preface to the First Edition</i> | <i>xiv</i> |
| <i>License Information</i> | <i>xvi</i> |
| <i>Computer Programs by Chapter and Section</i> | <i>xix</i> |
| 1 Preliminaries | 1 |
| 1.0 Introduction | 1 |
| 1.1 Program Organization and Control Structures | 5 |
| 1.2 Some C Conventions for Scientific Computing | 15 |
| 1.3 Error, Accuracy, and Stability | 28 |
| 2 Solution of Linear Algebraic Equations | 32 |
| 2.0 Introduction | 32 |
| 2.1 Gauss-Jordan Elimination | 36 |
| 2.2 Gaussian Elimination with Backsubstitution | 41 |
| 2.3 LU Decomposition and Its Applications | 43 |
| 2.4 Tridiagonal and Band Diagonal Systems of Equations | 50 |
| 2.5 Iterative Improvement of a Solution to Linear Equations | 55 |
| 2.6 Singular Value Decomposition | 59 |
| 2.7 Sparse Linear Systems | 71 |
| 2.8 Vandermonde Matrices and Toeplitz Matrices | 90 |
| 2.9 Cholesky Decomposition | 96 |
| 2.10 QR Decomposition | 98 |
| 2.11 Is Matrix Inversion an N^3 Process? | 102 |
| 3 Interpolation and Extrapolation | 105 |
| 3.0 Introduction | 105 |
| 3.1 Polynomial Interpolation and Extrapolation | 108 |
| 3.2 Rational Function Interpolation and Extrapolation | 111 |
| 3.3 Cubic Spline Interpolation | 113 |
| 3.4 How to Search an Ordered Table | 117 |
| 3.5 Coefficients of the Interpolating Polynomial | 120 |
| 3.6 Interpolation in Two or More Dimensions | 123 |

| | | |
|----------|--|------------|
| 4 | <i>Integration of Functions</i> | 129 |
| 4.0 | Introduction | 129 |
| 4.1 | Classical Formulas for Equally Spaced Abscissas | 130 |
| 4.2 | Elementary Algorithms | 136 |
| 4.3 | Romberg Integration | 140 |
| 4.4 | Improper Integrals | 141 |
| 4.5 | Gaussian Quadratures and Orthogonal Polynomials | 147 |
| 4.6 | Multidimensional Integrals | 161 |
| 5 | <i>Evaluation of Functions</i> | 165 |
| 5.0 | Introduction | 165 |
| 5.1 | Series and Their Convergence | 165 |
| 5.2 | Evaluation of Continued Fractions | 169 |
| 5.3 | Polynomials and Rational Functions | 173 |
| 5.4 | Complex Arithmetic | 176 |
| 5.5 | Recurrence Relations and Clenshaw's Recurrence Formula | 178 |
| 5.6 | Quadratic and Cubic Equations | 183 |
| 5.7 | Numerical Derivatives | 186 |
| 5.8 | Chebyshev Approximation | 190 |
| 5.9 | Derivatives or Integrals of a Chebyshev-approximated Function | 195 |
| 5.10 | Polynomial Approximation from Chebyshev Coefficients | 197 |
| 5.11 | Economization of Power Series | 198 |
| 5.12 | Padé Approximants | 200 |
| 5.13 | Rational Chebyshev Approximation | 204 |
| 5.14 | Evaluation of Functions by Path Integration | 208 |
| 6 | <i>Special Functions</i> | 212 |
| 6.0 | Introduction | 212 |
| 6.1 | Gamma Function, Beta Function, Factorials, Binomial Coefficients | 213 |
| 6.2 | Incomplete Gamma Function, Error Function, Chi-Square Probability Function, Cumulative Poisson Function | 216 |
| 6.3 | Exponential Integrals | 222 |
| 6.4 | Incomplete Beta Function, Student's Distribution, F-Distribution, Cumulative Binomial Distribution | 226 |
| 6.5 | Bessel Functions of Integer Order | 230 |
| 6.6 | Modified Bessel Functions of Integer Order | 236 |
| 6.7 | Bessel Functions of Fractional Order, Airy Functions, Spherical Bessel Functions | 240 |
| 6.8 | Spherical Harmonics | 252 |
| 6.9 | Fresnel Integrals, Cosine and Sine Integrals | 255 |
| 6.10 | Dawson's Integral | 259 |
| 6.11 | Elliptic Integrals and Jacobian Elliptic Functions | 261 |
| 6.12 | Hypergeometric Functions | 271 |
| 7 | <i>Random Numbers</i> | 274 |
| 7.0 | Introduction | 274 |
| 7.1 | Uniform Deviates | 275 |

| | |
|--|------------|
| 7.2 Transformation Method: Exponential and Normal Deviates | 287 |
| 7.3 Rejection Method: Gamma, Poisson, Binomial Deviates | 290 |
| 7.4 Generation of Random Bits | 296 |
| 7.5 Random Sequences Based on Data Encryption | 300 |
| 7.6 Simple Monte Carlo Integration | 304 |
| 7.7 Quasi- (that is, Sub-) Random Sequences | 309 |
| 7.8 Adaptive and Recursive Monte Carlo Methods | 316 |
| 8 Sorting | 329 |
| 8.0 Introduction | 329 |
| 8.1 Straight Insertion and Shell's Method | 330 |
| 8.2 Quicksort | 332 |
| 8.3 Heapsort | 336 |
| 8.4 Indexing and Ranking | 338 |
| 8.5 Selecting the M th Largest | 341 |
| 8.6 Determination of Equivalence Classes | 345 |
| 9 Root Finding and Nonlinear Sets of Equations | 347 |
| 9.0 Introduction | 347 |
| 9.1 Bracketing and Bisection | 350 |
| 9.2 Secant Method, False Position Method, and Ridders' Method | 354 |
| 9.3 Van Wijngaarden–Dekker–Brent Method | 359 |
| 9.4 Newton-Raphson Method Using Derivative | 362 |
| 9.5 Roots of Polynomials | 369 |
| 9.6 Newton-Raphson Method for Nonlinear Systems of Equations | 379 |
| 9.7 Globally Convergent Methods for Nonlinear Systems of Equations | 383 |
| 10 Minimization or Maximization of Functions | 394 |
| 10.0 Introduction | 394 |
| 10.1 Golden Section Search in One Dimension | 397 |
| 10.2 Parabolic Interpolation and Brent's Method in One Dimension | 402 |
| 10.3 One-Dimensional Search with First Derivatives | 405 |
| 10.4 Downhill Simplex Method in Multidimensions | 408 |
| 10.5 Direction Set (Powell's) Methods in Multidimensions | 412 |
| 10.6 Conjugate Gradient Methods in Multidimensions | 420 |
| 10.7 Variable Metric Methods in Multidimensions | 425 |
| 10.8 Linear Programming and the Simplex Method | 430 |
| 10.9 Simulated Annealing Methods | 444 |
| 11 Eigensystems | 456 |
| 11.0 Introduction | 456 |
| 11.1 Jacobi Transformations of a Symmetric Matrix | 463 |
| 11.2 Reduction of a Symmetric Matrix to Tridiagonal Form: Givens and Householder Reductions | 469 |
| 11.3 Eigenvalues and Eigenvectors of a Tridiagonal Matrix | 475 |
| 11.4 Hermitian Matrices | 481 |
| 11.5 Reduction of a General Matrix to Hessenberg Form | 482 |

| | |
|--|------------|
| 11.6 The QR Algorithm for Real Hessenberg Matrices | 486 |
| 11.7 Improving Eigenvalues and/or Finding Eigenvectors by Inverse Iteration | 493 |
| 12 Fast Fourier Transform | 496 |
| 12.0 Introduction | 496 |
| 12.1 Fourier Transform of Discretely Sampled Data | 500 |
| 12.2 Fast Fourier Transform (FFT) | 504 |
| 12.3 FFT of Real Functions, Sine and Cosine Transforms | 510 |
| 12.4 FFT in Two or More Dimensions | 521 |
| 12.5 Fourier Transforms of Real Data in Two and Three Dimensions | 525 |
| 12.6 External Storage or Memory-Local FFTs | 532 |
| 13 Fourier and Spectral Applications | 537 |
| 13.0 Introduction | 537 |
| 13.1 Convolution and Deconvolution Using the FFT | 538 |
| 13.2 Correlation and Autocorrelation Using the FFT | 545 |
| 13.3 Optimal (Wiener) Filtering with the FFT | 547 |
| 13.4 Power Spectrum Estimation Using the FFT | 549 |
| 13.5 Digital Filtering in the Time Domain | 558 |
| 13.6 Linear Prediction and Linear Predictive Coding | 564 |
| 13.7 Power Spectrum Estimation by the Maximum Entropy (All Poles) Method | 572 |
| 13.8 Spectral Analysis of Unevenly Sampled Data | 575 |
| 13.9 Computing Fourier Integrals Using the FFT | 584 |
| 13.10 Wavelet Transforms | 591 |
| 13.11 Numerical Use of the Sampling Theorem | 606 |
| 14 Statistical Description of Data | 609 |
| 14.0 Introduction | 609 |
| 14.1 Moments of a Distribution: Mean, Variance, Skewness, and So Forth | 610 |
| 14.2 Do Two Distributions Have the Same Means or Variances? | 615 |
| 14.3 Are Two Distributions Different? | 620 |
| 14.4 Contingency Table Analysis of Two Distributions | 628 |
| 14.5 Linear Correlation | 636 |
| 14.6 Nonparametric or Rank Correlation | 639 |
| 14.7 Do Two-Dimensional Distributions Differ? | 645 |
| 14.8 Savitzky-Golay Smoothing Filters | 650 |
| 15 Modeling of Data | 656 |
| 15.0 Introduction | 656 |
| 15.1 Least Squares as a Maximum Likelihood Estimator | 657 |
| 15.2 Fitting Data to a Straight Line | 661 |
| 15.3 Straight-Line Data with Errors in Both Coordinates | 666 |
| 15.4 General Linear Least Squares | 671 |
| 15.5 Nonlinear Models | 681 |

| | |
|---|------------|
| 15.6 Confidence Limits on Estimated Model Parameters | 689 |
| 15.7 Robust Estimation | 699 |
| 16 Integration of Ordinary Differential Equations | 707 |
| 16.0 Introduction | 707 |
| 16.1 Runge-Kutta Method | 710 |
| 16.2 Adaptive Stepsize Control for Runge-Kutta | 714 |
| 16.3 Modified Midpoint Method | 722 |
| 16.4 Richardson Extrapolation and the Bulirsch-Stoer Method | 724 |
| 16.5 Second-Order Conservative Equations | 732 |
| 16.6 Stiff Sets of Equations | 734 |
| 16.7 Multistep, Multivalued, and Predictor-Corrector Methods | 747 |
| 17 Two Point Boundary Value Problems | 753 |
| 17.0 Introduction | 753 |
| 17.1 The Shooting Method | 757 |
| 17.2 Shooting to a Fitting Point | 760 |
| 17.3 Relaxation Methods | 762 |
| 17.4 A Worked Example: Spheroidal Harmonics | 772 |
| 17.5 Automated Allocation of Mesh Points | 783 |
| 17.6 Handling Internal Boundary Conditions or Singular Points | 784 |
| 18 Integral Equations and Inverse Theory | 788 |
| 18.0 Introduction | 788 |
| 18.1 Fredholm Equations of the Second Kind | 791 |
| 18.2 Volterra Equations | 794 |
| 18.3 Integral Equations with Singular Kernels | 797 |
| 18.4 Inverse Problems and the Use of A Priori Information | 804 |
| 18.5 Linear Regularization Methods | 808 |
| 18.6 Backus-Gilbert Method | 815 |
| 18.7 Maximum Entropy Image Restoration | 818 |
| 19 Partial Differential Equations | 827 |
| 19.0 Introduction | 827 |
| 19.1 Flux-Conservative Initial Value Problems | 834 |
| 19.2 Diffusive Initial Value Problems | 847 |
| 19.3 Initial Value Problems in Multidimensions | 853 |
| 19.4 Fourier and Cyclic Reduction Methods for Boundary Value Problems | 857 |
| 19.5 Relaxation Methods for Boundary Value Problems | 863 |
| 19.6 Multigrid Methods for Boundary Value Problems | 871 |
| 20 Less-Numerical Algorithms | 889 |
| 20.0 Introduction | 889 |
| 20.1 Diagnosing Machine Parameters | 889 |
| 20.2 Gray Codes | 894 |

| | |
|---|------------|
| 20.3 Cyclic Redundancy and Other Checksums | 896 |
| 20.4 Huffman Coding and Compression of Data | 903 |
| 20.5 Arithmetic Coding | 910 |
| 20.6 Arithmetic at Arbitrary Precision | 915 |
| <i>References</i> | 926 |
| <i>Appendix A: Table of Prototype Declarations</i> | 930 |
| <i>Appendix B: Utility Routines</i> | 940 |
| <i>Appendix C: Complex Arithmetic</i> | 948 |
| <i>Index of Programs and Dependencies</i> | 951 |
| <i>General Index</i> | 965 |

Preface to the Second Edition

Our aim in writing the original edition of *Numerical Recipes* was to provide a book that combined general discussion, analytical mathematics, algorithmics, and actual working programs. The success of the first edition puts us now in a difficult, though hardly unenviable, position. We wanted, then and now, to write a book that is informal, fearlessly editorial, unesoteric, and above all useful. There is a danger that, if we are not careful, we might produce a second edition that is weighty, balanced, scholarly, and boring.

It is a mixed blessing that we know more now than we did six years ago. Then, we were making educated guesses, based on existing literature and our own research, about which numerical techniques were the most important and robust. Now, we have the benefit of direct feedback from a large reader community. Letters to our alter-ego enterprise, Numerical Recipes Software, are in the thousands per year. (Please, *don't telephone* us.) Our post office box has become a magnet for letters pointing out that we have omitted some particular technique, well known to be important in a particular field of science or engineering. We value such letters, and digest them carefully, especially when they point us to specific references in the literature.

The inevitable result of this input is that this Second Edition of *Numerical Recipes* is substantially larger than its predecessor, in fact about 50% larger both in words and number of included programs (the latter now numbering well over 300). “Don’t let the book grow in size,” is the advice that we received from several wise colleagues. We have tried to follow the intended spirit of that advice, even as we violate the letter of it. We have not lengthened, or increased in difficulty, the book’s principal discussions of mainstream topics. Many new topics are presented at this same accessible level. Some topics, both from the earlier edition and new to this one, are now set in smaller type that labels them as being “advanced.” The reader who ignores such advanced sections completely will not, we think, find any lack of continuity in the shorter volume that results.

Here are some highlights of the new material in this Second Edition:

- a new chapter on integral equations and inverse methods
- a detailed treatment of multigrid methods for solving elliptic partial differential equations
- routines for band diagonal linear systems
- improved routines for linear algebra on sparse matrices
- Cholesky and QR decomposition
- orthogonal polynomials and Gaussian quadratures for arbitrary weight functions
- methods for calculating numerical derivatives
- Padé approximants, and rational Chebyshev approximation
- Bessel functions, and modified Bessel functions, of fractional order; and several other new special functions
- improved random number routines
- quasi-random sequences
- routines for adaptive and recursive Monte Carlo integration in high-dimensional spaces
- globally convergent methods for sets of nonlinear equations

- simulated annealing minimization for continuous control spaces
- fast Fourier transform (FFT) for real data in two and three dimensions
- fast Fourier transform (FFT) using external storage
- improved fast cosine transform routines
- wavelet transforms
- Fourier integrals with upper and lower limits
- spectral analysis on unevenly sampled data
- Savitzky-Golay smoothing filters
- fitting straight line data with errors in both coordinates
- a two-dimensional Kolmogorov-Smirnoff test
- the statistical bootstrap method
- embedded Runge-Kutta-Fehlberg methods for differential equations
- high-order methods for stiff differential equations
- a new chapter on “less-numerical” algorithms, including Huffman and arithmetic coding, arbitrary precision arithmetic, and several other topics.

Consult the Preface to the First Edition, following, or the Table of Contents, for a list of the more “basic” subjects treated.

Acknowledgments

It is not possible for us to list by name here all the readers who have made useful suggestions; we are grateful for these. In the text, we attempt to give specific attribution for ideas that appear to be original, and not known in the literature. We apologize in advance for any omissions.

Some readers and colleagues have been particularly generous in providing us with ideas, comments, suggestions, and programs for this Second Edition. We especially want to thank George Rybicki, Philip Pinto, Peter Lepage, Robert Lupton, Douglas Eardley, Ramesh Narayan, David Spergel, Alan Oppenheim, Sallie Baliunas, Scott Tremaine, Glennys Farrar, Steven Block, John Peacock, Thomas Lored, Matthew Choptuik, Gregory Cook, L. Samuel Finn, P. Deuflhard, Harold Lewis, Peter Weinberger, David Syer, Richard Ferch, Steven Ebstein, Bradley Keister, and William Gould. We have been helped by Nancy Lee Snyder’s mastery of a complicated \TeX manuscript. We express appreciation to our editors Lauren Cowles and Alan Harvey at Cambridge University Press, and to our production editor Russell Hahn. We remain, of course, grateful to the individuals acknowledged in the Preface to the First Edition.

Special acknowledgment is due to programming consultant Seth Finkelstein, who wrote, rewrote, or influenced many of the routines in this book, as well as in its FORTRAN-language twin and the companion Example books. Our project has benefited enormously from Seth’s talent for detecting, and following the trail of, even very slight anomalies (often compiler bugs, but occasionally our errors), and from his good programming sense. To the extent that this edition of *Numerical Recipes in C* has a more graceful and “C-like” programming style than its predecessor, most of the credit goes to Seth. (Of course, we accept the blame for the FORTRANish lapses that still remain.)

We prepared this book for publication on DEC and Sun workstations running the UNIX operating system, and on a 486/33 PC compatible running MS-DOS 5.0/Windows 3.0. (See §1.0 for a list of additional computers used in

program tests.) We enthusiastically recommend the principal software used: GNU Emacs, T_EX, Perl, Adobe Illustrator, and PostScript. Also used were a variety of C compilers – too numerous (and sometimes too buggy) for individual acknowledgment. It is a sobering fact that our standard test suite (exercising all the routines in this book) has uncovered compiler bugs in many of the compilers tried. When possible, we work with developers to see that such bugs get fixed; we encourage interested compiler developers to contact us about such arrangements.

WHP and SAT acknowledge the continued support of the U.S. National Science Foundation for their research on computational methods. D.A.R.P.A. support is acknowledged for §13.10 on wavelets.

June, 1992

William H. Press
Saul A. Teukolsky
William T. Vetterling
Brian P. Flannery

Preface to the First Edition

We call this book *Numerical Recipes* for several reasons. In one sense, this book is indeed a “cookbook” on numerical computation. However there is an important distinction between a cookbook and a restaurant menu. The latter presents choices among complete dishes in each of which the individual flavors are blended and disguised. The former — and this book — reveals the individual ingredients and explains how they are prepared and combined.

Another purpose of the title is to connote an eclectic mixture of presentational techniques. This book is unique, we think, in offering, for each topic considered, a certain amount of general discussion, a certain amount of analytical mathematics, a certain amount of discussion of algorithmics, and (most important) actual implementations of these ideas in the form of working computer routines. Our task has been to find the right balance among these ingredients for each topic. You will find that for some topics we have tilted quite far to the analytic side; this where we have felt there to be gaps in the “standard” mathematical training. For other topics, where the mathematical prerequisites are universally held, we have tilted towards more in-depth discussion of the nature of the computational algorithms, or towards practical questions of implementation.

We admit, therefore, to some unevenness in the “level” of this book. About half of it is suitable for an advanced undergraduate course on numerical computation for science or engineering majors. The other half ranges from the level of a graduate course to that of a professional reference. Most cookbooks have, after all, recipes at varying levels of complexity. An attractive feature of this approach, we think, is that the reader can use the book at increasing levels of sophistication as his/her experience grows. Even inexperienced readers should be able to use our most advanced routines as black boxes. Having done so, we hope that these readers will subsequently go back and learn what secrets are inside.

If there is a single dominant theme in this book, it is that practical methods of numerical computation can be simultaneously efficient, clever, and — important — clear. The alternative viewpoint, that efficient computational methods must necessarily be so arcane and complex as to be useful only in “black box” form, we firmly reject.

Our purpose in this book is thus to open up a large number of computational black boxes to your scrutiny. We want to teach you to take apart these black boxes and to put them back together again, modifying them to suit your specific needs. We assume that you are mathematically literate, i.e., that you have the normal mathematical preparation associated with an undergraduate degree in a physical science, or engineering, or economics, or a quantitative social science. We assume that you know how to program a computer. We do not assume that you have any prior formal knowledge of numerical analysis or numerical methods.

The scope of *Numerical Recipes* is supposed to be “everything up to, but not including, partial differential equations.” We honor this in the breach: First, we *do* have one introductory chapter on methods for partial differential equations (Chapter 19). Second, we obviously cannot include *everything* else. All the so-called “standard” topics of a numerical analysis course have been included in this book:

linear equations (Chapter 2), interpolation and extrapolation (Chapter 3), integration (Chapter 4), nonlinear root-finding (Chapter 9), eigensystems (Chapter 11), and ordinary differential equations (Chapter 16). Most of these topics have been taken beyond their standard treatments into some advanced material which we have felt to be particularly important or useful.

Some other subjects that we cover in detail are not usually found in the standard numerical analysis texts. These include the evaluation of functions and of particular special functions of higher mathematics (Chapters 5 and 6); random numbers and Monte Carlo methods (Chapter 7); sorting (Chapter 8); optimization, including multidimensional methods (Chapter 10); Fourier transform methods, including FFT methods and other spectral methods (Chapters 12 and 13); two chapters on the statistical description and modeling of data (Chapters 14 and 15); and two-point boundary value problems, both shooting and relaxation methods (Chapter 17).

The programs in this book are included in ANSI-standard C. Versions of the book in FORTRAN, Pascal, and BASIC are available separately. We have more to say about the C language, and the computational environment assumed by our routines, in §1.1 (Introduction).

Acknowledgments

Many colleagues have been generous in giving us the benefit of their numerical and computational experience, in providing us with programs, in commenting on the manuscript, or in general encouragement. We particularly wish to thank George Rybicki, Douglas Eardley, Philip Marcus, Stuart Shapiro, Paul Horowitz, Bruce Musicus, Irwin Shapiro, Stephen Wolfram, Henry Abarbanel, Larry Smarr, Richard Muller, John Bahcall, and A.G.W. Cameron.

We also wish to acknowledge two individuals whom we have never met: Forman Acton, whose 1970 textbook *Numerical Methods that Work* (New York: Harper and Row) has surely left its stylistic mark on us; and Donald Knuth, both for his series of books on *The Art of Computer Programming* (Reading, MA: Addison-Wesley), and for T_EX, the computer typesetting language which immensely aided production of this book.

Research by the authors on computational methods was supported in part by the U.S. National Science Foundation.

October, 1985

William H. Press
Brian P. Flannery
Saul A. Teukolsky
William T. Vetterling

License Information

Read this section if you want to use the programs in this book on a computer. You'll need to read the following Disclaimer of Warranty, get the programs onto your computer, and acquire a Numerical Recipes software license. (Without this license, which can be the free "immediate license" under terms described below, the book is intended as a text and reference book, for reading purposes only.)

Disclaimer of Warranty

We make no warranties, express or implied, that the programs contained in this volume are free of error, or are consistent with any particular standard of merchantability, or that they will meet your requirements for any particular application. They should not be relied on for solving a problem whose incorrect solution could result in injury to a person or loss of property. If you do use the programs in such a manner, it is at your own risk. The authors and publisher disclaim all liability for direct or consequential damages resulting from your use of the programs.

How to Get the Code onto Your Computer

Pick one of the following methods:

- You can type the programs from this book directly into your computer. In this case, the *only* kind of license available to you is the free "immediate license" (see below). You are not authorized to transfer or distribute a machine-readable copy to any other person, nor to have any other person type the programs into a computer on your behalf. We do not want to hear bug reports from you if you choose this option, because experience has shown that *virtually all* reported bugs in such cases are typing errors!
- You can download the Numerical Recipes programs electronically from the Numerical Recipes On-Line Software Store, located at <http://www.nr.com>, our Web site. All the files (Recipes and demonstration programs) are packaged as a single compressed file. You'll need to purchase a license to download and unpack them. Any number of single-screen licenses can be purchased instantly (with discount for multiple screens) from the On-Line Store, with fees that depend on your operating system (Windows or Macintosh versus Linux or UNIX) and whether you are affiliated with an educational institution. Purchasing a single-screen license is also the way to start if you want to acquire a more general (site or corporate) license; your single-screen cost will be subtracted from the cost of any later license upgrade.
- You can purchase media containing the programs from Cambridge University Press. A CD-ROM version in ISO-9660 format for Windows and Macintosh systems contains the complete C software, and also the C++ version. More extensive CD-ROMs in ISO-9660 format for Windows, Macintosh, and UNIX/Linux systems are also available; these include the C, C++, and Fortran versions on a single CD-ROM (as well as versions in Pascal and BASIC from the first edition). These CD-ROMs are available with a single-screen license for Windows or Macintosh (order ISBN 0 521 750350), or (at a slightly higher price) with a single-screen license for UNIX/Linux workstations (order ISBN 0 521 750369). Orders for media from Cambridge University Press can be placed at 800 872-7423 (North America only) or by email to orders@cup.org (North America) or directcustserv@cambridge.org (rest of world). Or, visit the Web site <http://www.cambridge.org>.

Types of License Offered

Here are the types of licenses that we offer. Note that some types are automatically acquired with the purchase of media from Cambridge University Press, or of an unlocking password from the Numerical Recipes On-Line Software Store, while other types of licenses require that you communicate specifically with Numerical Recipes Software (email: orders@nr.com or fax: 781 863-1739). Our Web site <http://www.nr.com> has additional information.

- [“Immediate License”] If you are the individual owner of a copy of this book and you type one or more of its routines into your computer, we authorize you to use them on that computer for your own personal and noncommercial purposes. You are not authorized to transfer or distribute machine-readable copies to any other person, or to use the routines on more than one machine, or to distribute executable programs containing our routines. This is the only free license.
- [“Single-Screen License”] This is the most common type of low-cost license, with terms governed by our Single Screen (Shrinkwrap) License document (complete terms available through our Web site). Basically, this license lets you use Numerical Recipes routines on any one screen (PC, workstation, X-terminal, etc.). You may also, under this license, transfer pre-compiled, executable programs incorporating our routines to other, unlicensed, screens or computers, providing that (i) your application is noncommercial (i.e., does not involve the selling of your program for a fee), (ii) the programs were first developed, compiled, and successfully run on a licensed screen, and (iii) our routines are bound into the programs in such a manner that they cannot be accessed as individual routines and cannot practically be unbound and used in other programs. That is, under this license, your program user must not be able to use our programs as part of a program library or “mix-and-match” workbench. Conditions for other types of commercial or noncommercial distribution may be found on our Web site (<http://www.nr.com>).
- [“Multi-Screen, Server, Site, and Corporate Licenses”] The terms of the Single Screen License can be extended to designated groups of machines, defined by number of screens, number of machines, locations, or ownership. Significant discounts from the corresponding single-screen prices are available when the estimated number of screens exceeds 40. Contact Numerical Recipes Software (email: orders@nr.com or fax: 781 863-1739) for details.
- [“Course Right-to-Copy License”] Instructors at accredited educational institutions who have adopted this book for a course, and who have already purchased a Single Screen License (either acquired with the purchase of media, or from the Numerical Recipes On-Line Software Store), may license the programs for use in that course as follows: Mail your name, title, and address; the course name, number, dates, and estimated enrollment; and advance payment of \$5 per (estimated) student to Numerical Recipes Software, at this address: P.O. Box 243, Cambridge, MA 02238 (USA). You will receive by return mail a license authorizing you to make copies of the programs for use by your students, and/or to transfer the programs to a machine accessible to your students (but only for the duration of the course).

About Copyrights on Computer Programs

Like artistic or literary compositions, computer programs are protected by copyright. Generally it is an infringement for you to copy into your computer a program from a copyrighted source. (It is also not a friendly thing to do, since it deprives the program’s author of compensation for his or her creative effort.) Under

copyright law, all “derivative works” (modified versions, or translations into another computer language) also come under the same copyright as the original work.

Copyright does not protect ideas, but only the expression of those ideas in a particular form. In the case of a computer program, the ideas consist of the program’s methodology and algorithm, including the necessary sequence of steps adopted by the programmer. The expression of those ideas is the program source code (particularly any arbitrary or stylistic choices embodied in it), its derived object code, and any other derivative works.

If you analyze the ideas contained in a program, and then express those ideas in your own completely different implementation, then that new program implementation belongs to you. That is what we have done for those programs in this book that are not entirely of our own devising. When programs in this book are said to be “based” on programs published in copyright sources, we mean that the ideas are the same. The expression of these ideas as source code is our own. We believe that no material in this book infringes on an existing copyright.

Trademarks

Several registered trademarks appear within the text of this book: Sun is a trademark of Sun Microsystems, Inc. SPARC and SPARCstation are trademarks of SPARC International, Inc. Microsoft, Windows 95, Windows NT, PowerStation, and MS are trademarks of Microsoft Corporation. DEC, VMS, Alpha AXP, and ULTRIX are trademarks of Digital Equipment Corporation. IBM is a trademark of International Business Machines Corporation. Apple and Macintosh are trademarks of Apple Computer, Inc. UNIX is a trademark licensed exclusively through X/Open Co. Ltd. IMSL is a trademark of Visual Numerics, Inc. NAG refers to proprietary computer software of Numerical Algorithms Group (USA) Inc. PostScript and Adobe Illustrator are trademarks of Adobe Systems Incorporated. Last, and no doubt least, Numerical Recipes (when identifying products) is a trademark of Numerical Recipes Software.

Attributions

The fact that ideas are legally “free as air” in no way supersedes the ethical requirement that ideas be credited to their known originators. When programs in this book are based on known sources, whether copyrighted or in the public domain, published or “handed-down,” we have attempted to give proper attribution. Unfortunately, the lineage of many programs in common circulation is often unclear. We would be grateful to readers for new or corrected information regarding attributions, which we will attempt to incorporate in subsequent printings.

Computer Programs by Chapter and Section

| | | |
|------|---------|--|
| 1.0 | flmoon | calculate phases of the moon by date |
| 1.1 | julday | Julian Day number from calendar date |
| 1.1 | badluk | Friday the 13th when the moon is full |
| 1.1 | caldat | calendar date from Julian day number |
| 2.1 | gaussj | Gauss-Jordan matrix inversion and linear equation solution |
| 2.3 | ludcmp | linear equation solution, LU decomposition |
| 2.3 | lubksb | linear equation solution, backsubstitution |
| 2.4 | tridag | solution of tridiagonal systems |
| 2.4 | banmul | multiply vector by band diagonal matrix |
| 2.4 | bandec | band diagonal systems, decomposition |
| 2.4 | banbks | band diagonal systems, backsubstitution |
| 2.5 | mprove | linear equation solution, iterative improvement |
| 2.6 | svbksb | singular value backsubstitution |
| 2.6 | svdcmp | singular value decomposition of a matrix |
| 2.6 | pythag | calculate $(a^2 + b^2)^{1/2}$ without overflow |
| 2.7 | cyclic | solution of cyclic tridiagonal systems |
| 2.7 | sprsin | convert matrix to sparse format |
| 2.7 | spr sax | product of sparse matrix and vector |
| 2.7 | sprstx | product of transpose sparse matrix and vector |
| 2.7 | sprstp | transpose of sparse matrix |
| 2.7 | sprspm | pattern multiply two sparse matrices |
| 2.7 | sprstm | threshold multiply two sparse matrices |
| 2.7 | linbcg | biconjugate gradient solution of sparse systems |
| 2.7 | snrm | used by linbcg for vector norm |
| 2.7 | atimes | used by linbcg for sparse multiplication |
| 2.7 | asolve | used by linbcg for preconditioner |
| 2.8 | vander | solve Vandermonde systems |
| 2.8 | toeplz | solve Toeplitz systems |
| 2.9 | choldc | Cholesky decomposition |
| 2.9 | cholsl | Cholesky backsubstitution |
| 2.10 | qrdcmp | QR decomposition |
| 2.10 | qrsolv | QR backsubstitution |
| 2.10 | rsolv | right triangular backsubstitution |
| 2.10 | grupdt | update a QR decomposition |
| 2.10 | rotate | Jacobi rotation used by grupdt |
| 3.1 | polint | polynomial interpolation |
| 3.2 | ratint | rational function interpolation |
| 3.3 | spline | construct a cubic spline |
| 3.3 | splint | cubic spline interpolation |
| 3.4 | locate | search an ordered table by bisection |

| | | |
|------|--------|---|
| 3.4 | hunt | search a table when calls are correlated |
| 3.5 | polcoe | polynomial coefficients from table of values |
| 3.5 | polcof | polynomial coefficients from table of values |
| 3.6 | polin2 | two-dimensional polynomial interpolation |
| 3.6 | bcucof | construct two-dimensional bicubic |
| 3.6 | bcuint | two-dimensional bicubic interpolation |
| 3.6 | splie2 | construct two-dimensional spline |
| 3.6 | splin2 | two-dimensional spline interpolation |
| 4.2 | trapzd | trapezoidal rule |
| 4.2 | qtrap | integrate using trapezoidal rule |
| 4.2 | qsimp | integrate using Simpson's rule |
| 4.3 | qromb | integrate using Romberg adaptive method |
| 4.4 | midpnt | extended midpoint rule |
| 4.4 | qromo | integrate using open Romberg adaptive method |
| 4.4 | midinf | integrate a function on a semi-infinite interval |
| 4.4 | midsql | integrate a function with lower square-root singularity |
| 4.4 | midsqu | integrate a function with upper square-root singularity |
| 4.4 | midexp | integrate a function that decreases exponentially |
| 4.5 | qgaus | integrate a function by Gaussian quadratures |
| 4.5 | gauleg | Gauss-Legendre weights and abscissas |
| 4.5 | gaulag | Gauss-Laguerre weights and abscissas |
| 4.5 | gauher | Gauss-Hermite weights and abscissas |
| 4.5 | gaujac | Gauss-Jacobi weights and abscissas |
| 4.5 | gaucof | quadrature weights from orthogonal polynomials |
| 4.5 | orthog | construct nonclassical orthogonal polynomials |
| 4.6 | quad3d | integrate a function over a three-dimensional space |
| 5.1 | eulsum | sum a series by Euler-van Wijngaarden algorithm |
| 5.3 | ddpoly | evaluate a polynomial and its derivatives |
| 5.3 | poldiv | divide one polynomial by another |
| 5.3 | ratval | evaluate a rational function |
| 5.7 | dfridr | numerical derivative by Ridders' method |
| 5.8 | chebft | fit a Chebyshev polynomial to a function |
| 5.8 | chebev | Chebyshev polynomial evaluation |
| 5.9 | chder | derivative of a function already Chebyshev fitted |
| 5.9 | chint | integrate a function already Chebyshev fitted |
| 5.10 | chebpc | polynomial coefficients from a Chebyshev fit |
| 5.10 | pcshft | polynomial coefficients of a shifted polynomial |
| 5.11 | pccheb | inverse of chebpc; use to economize power series |
| 5.12 | pade | Padé approximant from power series coefficients |
| 5.13 | ratlsq | rational fit by least-squares method |
| 6.1 | gammln | logarithm of gamma function |
| 6.1 | factrl | factorial function |
| 6.1 | bico | binomial coefficients function |
| 6.1 | factln | logarithm of factorial function |

| | | |
|------|--------|--|
| 6.1 | beta | beta function |
| 6.2 | gamm | incomplete gamma function |
| 6.2 | gammq | complement of incomplete gamma function |
| 6.2 | gser | series used by gamm and gammq |
| 6.2 | gcf | continued fraction used by gamm and gammq |
| 6.2 | erff | error function |
| 6.2 | erffc | complementary error function |
| 6.2 | erfcc | complementary error function, concise routine |
| 6.3 | expint | exponential integral E_n |
| 6.3 | ei | exponential integral E_i |
| 6.4 | betai | incomplete beta function |
| 6.4 | betacf | continued fraction used by betai |
| 6.5 | bessj0 | Bessel function J_0 |
| 6.5 | bessy0 | Bessel function Y_0 |
| 6.5 | bessj1 | Bessel function J_1 |
| 6.5 | bessy1 | Bessel function Y_1 |
| 6.5 | bessy | Bessel function Y of general integer order |
| 6.5 | bessj | Bessel function J of general integer order |
| 6.6 | bessi0 | modified Bessel function I_0 |
| 6.6 | bessk0 | modified Bessel function K_0 |
| 6.6 | bessi1 | modified Bessel function I_1 |
| 6.6 | bessk1 | modified Bessel function K_1 |
| 6.6 | bessk | modified Bessel function K of integer order |
| 6.6 | bessi | modified Bessel function I of integer order |
| 6.7 | bessjy | Bessel functions of fractional order |
| 6.7 | beschb | Chebyshev expansion used by bessjy |
| 6.7 | bessik | modified Bessel functions of fractional order |
| 6.7 | airy | Airy functions |
| 6.7 | sphbes | spherical Bessel functions j_n and y_n |
| 6.8 | plgndr | Legendre polynomials, associated (spherical harmonics) |
| 6.9 | frenel | Fresnel integrals $S(x)$ and $C(x)$ |
| 6.9 | cisi | cosine and sine integrals Ci and Si |
| 6.10 | dawson | Dawson's integral |
| 6.11 | rf | Carlson's elliptic integral of the first kind |
| 6.11 | rd | Carlson's elliptic integral of the second kind |
| 6.11 | rj | Carlson's elliptic integral of the third kind |
| 6.11 | rc | Carlson's degenerate elliptic integral |
| 6.11 | ellf | Legendre elliptic integral of the first kind |
| 6.11 | elle | Legendre elliptic integral of the second kind |
| 6.11 | ellpi | Legendre elliptic integral of the third kind |
| 6.11 | sncndn | Jacobian elliptic functions |
| 6.12 | hypgeo | complex hypergeometric function |
| 6.12 | hypser | complex hypergeometric function, series evaluation |
| 6.12 | hypdrv | complex hypergeometric function, derivative of |
| 7.1 | ran0 | random deviate by Park and Miller minimal standard |
| 7.1 | ran1 | random deviate, minimal standard plus shuffle |

| | | |
|-----|--------|---|
| 7.1 | ran2 | random deviate by L'Ecuyer long period plus shuffle |
| 7.1 | ran3 | random deviate by Knuth subtractive method |
| 7.2 | expdev | exponential random deviates |
| 7.2 | gasdev | normally distributed random deviates |
| 7.3 | gamdev | gamma-law distribution random deviates |
| 7.3 | poidev | Poisson distributed random deviates |
| 7.3 | bnldev | binomial distributed random deviates |
| 7.4 | irbit1 | random bit sequence |
| 7.4 | irbit2 | random bit sequence |
| 7.5 | psdes | "pseudo-DES" hashing of 64 bits |
| 7.5 | ran4 | random deviates from DES-like hashing |
| 7.7 | sobseq | Sobol's quasi-random sequence |
| 7.8 | vegas | adaptive multidimensional Monte Carlo integration |
| 7.8 | rebin | sample rebinning used by vegas |
| 7.8 | miser | recursive multidimensional Monte Carlo integration |
| 7.8 | ranpt | get random point, used by miser |
| | | |
| 8.1 | piksr1 | sort an array by straight insertion |
| 8.1 | piksr2 | sort two arrays by straight insertion |
| 8.1 | shell | sort an array by Shell's method |
| 8.2 | sort | sort an array by quicksort method |
| 8.2 | sort2 | sort two arrays by quicksort method |
| 8.3 | hpsort | sort an array by heapsort method |
| 8.4 | indexx | construct an index for an array |
| 8.4 | sort3 | sort, use an index to sort 3 or more arrays |
| 8.4 | rank | construct a rank table for an array |
| 8.5 | select | find the N th largest in an array |
| 8.5 | selip | find the N th largest, without altering an array |
| 8.5 | hpsel | find M largest values, without altering an array |
| 8.6 | eclass | determine equivalence classes from list |
| 8.6 | eclazz | determine equivalence classes from procedure |
| | | |
| 9.0 | scrsho | graph a function to search for roots |
| 9.1 | zbrac | outward search for brackets on roots |
| 9.1 | zbrak | inward search for brackets on roots |
| 9.1 | rtbis | find root of a function by bisection |
| 9.2 | rtflsp | find root of a function by false-position |
| 9.2 | rtsec | find root of a function by secant method |
| 9.2 | zridr | find root of a function by Ridders' method |
| 9.3 | zbrent | find root of a function by Brent's method |
| 9.4 | rtnewt | find root of a function by Newton-Raphson |
| 9.4 | rtsafe | find root of a function by Newton-Raphson and bisection |
| 9.5 | laguer | find a root of a polynomial by Laguerre's method |
| 9.5 | zroots | roots of a polynomial by Laguerre's method with deflation |
| | | |
| 9.5 | zrhqr | roots of a polynomial by eigenvalue methods |
| 9.5 | qroot | complex or double root of a polynomial, Bairstow |

| | | |
|------|---------|---|
| 9.6 | mnewt | Newton's method for systems of equations |
| 9.7 | lnsrch | search along a line, used by newt |
| 9.7 | newt | globally convergent multi-dimensional Newton's method |
| 9.7 | fdjac | finite-difference Jacobian, used by newt |
| 9.7 | fmin | norm of a vector function, used by newt |
| 9.7 | broydn | secant method for systems of equations |
| | | |
| 10.1 | mbrak | bracket the minimum of a function |
| 10.1 | golden | find minimum of a function by golden section search |
| 10.2 | brent | find minimum of a function by Brent's method |
| 10.3 | dbrent | find minimum of a function using derivative information |
| 10.4 | amoeba | minimize in N -dimensions by downhill simplex method |
| 10.4 | amotry | evaluate a trial point, used by amoeba |
| 10.5 | powell | minimize in N -dimensions by Powell's method |
| 10.5 | linmin | minimum of a function along a ray in N -dimensions |
| 10.5 | f1dim | function used by linmin |
| 10.6 | frprmn | minimize in N -dimensions by conjugate gradient |
| 10.6 | dlinmin | minimum of a function along a ray using derivatives |
| 10.6 | df1dim | function used by dlinmin |
| 10.7 | dfpmin | minimize in N -dimensions by variable metric method |
| 10.8 | simplx | linear programming maximization of a linear function |
| 10.8 | simpl1 | linear programming, used by simplx |
| 10.8 | simpl2 | linear programming, used by simplx |
| 10.8 | simpl3 | linear programming, used by simplx |
| 10.9 | anneal | traveling salesman problem by simulated annealing |
| 10.9 | revcst | cost of a reversal, used by anneal |
| 10.9 | reverse | do a reversal, used by anneal |
| 10.9 | trncst | cost of a transposition, used by anneal |
| 10.9 | trnspt | do a transposition, used by anneal |
| 10.9 | metrop | Metropolis algorithm, used by anneal |
| 10.9 | amebsa | simulated annealing in continuous spaces |
| 10.9 | amotsa | evaluate a trial point, used by amebse |
| | | |
| 11.1 | jacobi | eigenvalues and eigenvectors of a symmetric matrix |
| 11.1 | eigsrt | eigenvectors, sorts into order by eigenvalue |
| 11.2 | tred2 | Householder reduction of a real, symmetric matrix |
| 11.3 | tqli | eigensolution of a symmetric tridiagonal matrix |
| 11.5 | balanc | balance a nonsymmetric matrix |
| 11.5 | elmhes | reduce a general matrix to Hessenberg form |
| 11.6 | hqr | eigenvalues of a Hessenberg matrix |
| | | |
| 12.2 | four1 | fast Fourier transform (FFT) in one dimension |
| 12.3 | twofft | fast Fourier transform of two real functions |
| 12.3 | realft | fast Fourier transform of a single real function |
| 12.3 | sinf | fast sine transform |
| 12.3 | cosft1 | fast cosine transform with endpoints |
| 12.3 | cosft2 | "staggered" fast cosine transform |

| | | |
|-------|--------|--|
| 12.4 | fourn | fast Fourier transform in multidimensions |
| 12.5 | rlft3 | FFT of real data in two or three dimensions |
| 12.6 | fourfs | FFT for huge data sets on external media |
| 12.6 | fourew | rewind and permute files, used by fourfs |
| | | |
| 13.1 | convlv | convolution or deconvolution of data using FFT |
| 13.2 | correl | correlation or autocorrelation of data using FFT |
| 13.4 | spctrm | power spectrum estimation using FFT |
| 13.6 | memcof | evaluate maximum entropy (MEM) coefficients |
| 13.6 | fixrts | reflect roots of a polynomial into unit circle |
| 13.6 | predic | linear prediction using MEM coefficients |
| 13.7 | evlmem | power spectral estimation from MEM coefficients |
| 13.8 | period | power spectrum of unevenly sampled data |
| 13.8 | fasper | power spectrum of unevenly sampled larger data sets |
| 13.8 | spread | extrapolate value into array, used by fasper |
| 13.9 | dftcor | compute endpoint corrections for Fourier integrals |
| 13.9 | dftint | high-accuracy Fourier integrals |
| 13.10 | wt1 | one-dimensional discrete wavelet transform |
| 13.10 | daub4 | Daubechies 4-coefficient wavelet filter |
| 13.10 | pwtset | initialize coefficients for pwt |
| 13.10 | pwt | partial wavelet transform |
| 13.10 | wtn | multidimensional discrete wavelet transform |
| | | |
| 14.1 | moment | calculate moments of a data set |
| 14.2 | ttest | Student's t -test for difference of means |
| 14.2 | avevar | calculate mean and variance of a data set |
| 14.2 | tutest | Student's t -test for means, case of unequal variances |
| 14.2 | tptest | Student's t -test for means, case of paired data |
| 14.2 | ftest | F -test for difference of variances |
| 14.3 | chsone | chi-square test for difference between data and model |
| 14.3 | chstwo | chi-square test for difference between two data sets |
| 14.3 | ksone | Kolmogorov-Smirnov test of data against model |
| 14.3 | kstwo | Kolmogorov-Smirnov test between two data sets |
| 14.3 | probks | Kolmogorov-Smirnov probability function |
| 14.4 | cntab1 | contingency table analysis using chi-square |
| 14.4 | cntab2 | contingency table analysis using entropy measure |
| 14.5 | pearsn | Pearson's correlation between two data sets |
| 14.6 | spear | Spearman's rank correlation between two data sets |
| 14.6 | crank | replaces array elements by their rank |
| 14.6 | kendl1 | correlation between two data sets, Kendall's tau |
| 14.6 | kendl2 | contingency table analysis using Kendall's tau |
| 14.7 | ks2d1s | K-S test in two dimensions, data vs. model |
| 14.7 | quadct | count points by quadrants, used by ks2d1s |
| 14.7 | quadv1 | quadrant probabilities, used by ks2d1s |
| 14.7 | ks2d2s | K-S test in two dimensions, data vs. data |
| 14.8 | savgol | Savitzky-Golay smoothing coefficients |

| | | |
|------|---------------------|--|
| 15.2 | <code>fit</code> | least-squares fit data to a straight line |
| 15.3 | <code>fitexy</code> | fit data to a straight line, errors in both x and y |
| 15.3 | <code>chixy</code> | used by <code>fitexy</code> to calculate a χ^2 |
| 15.4 | <code>lfit</code> | general linear least-squares fit by normal equations |
| 15.4 | <code>covsrt</code> | rearrange covariance matrix, used by <code>lfit</code> |
| 15.4 | <code>svdfit</code> | linear least-squares fit by singular value decomposition |
| 15.4 | <code>svdvar</code> | variances from singular value decomposition |
| 15.4 | <code>fpoly</code> | fit a polynomial using <code>lfit</code> or <code>svdfit</code> |
| 15.4 | <code>fleg</code> | fit a Legendre polynomial using <code>lfit</code> or <code>svdfit</code> |
| 15.5 | <code>mrqmin</code> | nonlinear least-squares fit, Marquardt's method |
| 15.5 | <code>mrqcof</code> | used by <code>mrqmin</code> to evaluate coefficients |
| 15.5 | <code>fgauss</code> | fit a sum of Gaussians using <code>mrqmin</code> |
| 15.7 | <code>medfit</code> | fit data to a straight line robustly, least absolute deviation |
| 15.7 | <code>rofunc</code> | fit data robustly, used by <code>medfit</code> |
| | | |
| 16.1 | <code>rk4</code> | integrate one step of ODEs, fourth-order Runge-Kutta |
| 16.1 | <code>rkdumb</code> | integrate ODEs by fourth-order Runge-Kutta |
| 16.2 | <code>rkqs</code> | integrate one step of ODEs with accuracy monitoring |
| 16.2 | <code>rkck</code> | Cash-Karp-Runge-Kutta step used by <code>rkqs</code> |
| 16.2 | <code>odeint</code> | integrate ODEs with accuracy monitoring |
| 16.3 | <code>mmid</code> | integrate ODEs by modified midpoint method |
| 16.4 | <code>bsstep</code> | integrate ODEs, Bulirsch-Stoer step |
| 16.4 | <code>pzextr</code> | polynomial extrapolation, used by <code>bsstep</code> |
| 16.4 | <code>rzextr</code> | rational function extrapolation, used by <code>bsstep</code> |
| 16.5 | <code>stoerm</code> | integrate conservative second-order ODEs |
| 16.6 | <code>stiff</code> | integrate stiff ODEs by fourth-order Rosenbrock |
| 16.6 | <code>jacobn</code> | sample Jacobian routine for <code>stiff</code> |
| 16.6 | <code>derivs</code> | sample derivatives routine for <code>stiff</code> |
| 16.6 | <code>simpr</code> | integrate stiff ODEs by semi-implicit midpoint rule |
| 16.6 | <code>stifbs</code> | integrate stiff ODEs, Bulirsch-Stoer step |
| | | |
| 17.1 | <code>shoot</code> | solve two point boundary value problem by shooting |
| 17.2 | <code>shootf</code> | ditto, by shooting to a fitting point |
| 17.3 | <code>solvde</code> | two point boundary value problem, solve by relaxation |
| 17.3 | <code>bksub</code> | backsubstitution, used by <code>solvde</code> |
| 17.3 | <code>pinvs</code> | diagonalize a sub-block, used by <code>solvde</code> |
| 17.3 | <code>red</code> | reduce columns of a matrix, used by <code>solvde</code> |
| 17.4 | <code>sfroid</code> | spheroidal functions by method of <code>solvde</code> |
| 17.4 | <code>difeq</code> | spheroidal matrix coefficients, used by <code>sfroid</code> |
| 17.4 | <code>sphoot</code> | spheroidal functions by method of <code>shoot</code> |
| 17.4 | <code>sphfpt</code> | spheroidal functions by method of <code>shootf</code> |
| | | |
| 18.1 | <code>fred2</code> | solve linear Fredholm equations of the second kind |
| 18.1 | <code>fredin</code> | interpolate solutions obtained with <code>fred2</code> |
| 18.2 | <code>voltra</code> | linear Volterra equations of the second kind |
| 18.3 | <code>wgghts</code> | quadrature weights for an arbitrarily singular kernel |
| 18.3 | <code>kermom</code> | sample routine for moments of a singular kernel |

| | | |
|------|--------|--|
| 18.3 | quadmx | sample routine for a quadrature matrix |
| 18.3 | fredex | example of solving a singular Fredholm equation |
| 19.5 | sor | elliptic PDE solved by successive overrelaxation method |
| 19.6 | mglin | linear elliptic PDE solved by multigrid method |
| 19.6 | rstrct | half-weighting restriction, used by mglin, mgfas |
| 19.6 | interp | bilinear prolongation, used by mglin, mgfas |
| 19.6 | addint | interpolate and add, used by mglin |
| 19.6 | slvsm1 | solve on coarsest grid, used by mglin |
| 19.6 | relax | Gauss-Seidel relaxation, used by mglin |
| 19.6 | resid | calculate residual, used by mglin |
| 19.6 | copy | utility used by mglin, mgfas |
| 19.6 | fill0 | utility used by mglin |
| 19.6 | mgfas | nonlinear elliptic PDE solved by multigrid method |
| 19.6 | relax2 | Gauss-Seidel relaxation, used by mgfas |
| 19.6 | slvsm2 | solve on coarsest grid, used by mgfas |
| 19.6 | lop | applies nonlinear operator, used by mgfas |
| 19.6 | matadd | utility used by mgfas |
| 19.6 | matsub | utility used by mgfas |
| 19.6 | anorm2 | utility used by mgfas |
| 20.1 | machar | diagnose computer's floating arithmetic |
| 20.2 | igray | Gray code and its inverse |
| 20.3 | icrc1 | cyclic redundancy checksum, used by icrc |
| 20.3 | icrc | cyclic redundancy checksum |
| 20.3 | decchk | decimal check digit calculation or verification |
| 20.4 | hufmak | construct a Huffman code |
| 20.4 | hufapp | append bits to a Huffman code, used by hufmak |
| 20.4 | hufenc | use Huffman code to encode and compress a character |
| 20.4 | hufdec | use Huffman code to decode and decompress a character |
| 20.5 | arcmak | construct an arithmetic code |
| 20.5 | arcode | encode or decode a character using arithmetic coding |
| 20.5 | arcsum | add integer to byte string, used by arcode |
| 20.6 | mpops | multiple precision arithmetic, simpler operations |
| 20.6 | mpmul | multiple precision multiply, using FFT methods |
| 20.6 | mpinv | multiple precision reciprocal |
| 20.6 | mpdiv | multiple precision divide and remainder |
| 20.6 | mpsqr | multiple precision square root |
| 20.6 | mp2dfr | multiple precision conversion to decimal base |
| 20.6 | mppi | multiple precision example, compute many digits of π |