Numerical Recipes
in Fortran 77
Second Edition

Volume 1 of
Fortran Numerical Recipes

Numerical Recipes
in Fortran 77
The Art of Scientific Computing
Second Edition

Volume 1 of
Fortran Numerical Recipes

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

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
10 Stamford Road, Oakleigh, Melbourne 3166, Australia

First Edition originally published 1986; Second Edition originally published 1992 as
Numerical Recipes in FORTRAN: The Art of Scientific Computing
Reprinted with corrections, 1993, 1994, 1995.
Reprinted with corrections, 1996, 1997, 2001, as *Numerical Recipes in Fortran 77: The Art of Scientific Computing* (Vol. 1 of Fortran Numerical Recipes)

This reprinting is corrected to software version 2.10

Printed in the United States of America
Typeset in TeX

Library of Congress Cataloging in Publication Data

Numerical recipes in Fortran 77 : the art of scientific computing / William H. Press
. . . [et al.]. – 2nd ed.

Includes bibliographical references (p.) and index.

ISBN 0-521-43064-X

1. Numerical analysis–Computer programs. 2. Science–Mathematics–Computer programs.
3. FORTRAN (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 43064 X Volume 1 (this book)
ISBN 0 521 57439 0 Volume 2
ISBN 0 521 43721 0 Example book in FORTRAN
ISBN 0 521 57440 4 FORTRAN diskette (IBM 3.5″)
ISBN 0 521 57608 3 CDROM (IBM PC/Macintosh)
ISBN 0 521 57607 5 CDROM (UNIX)

Contents

Contents of Volume 2: Numerical Recipes in Fortran 90

Plan of the Two-Volume Edition

Fortran, long the epitome of stability, is once again a language in flux. Fortran 90 is not just the long-awaited updating of traditional Fortran 77 to modern computing practices, but also demonstrates Fortran's decisive bid to be the language of choice for parallel programming on multiprocessor computers.

At the same time, Fortran 90 is completely backwards-compatible with all Fortran 77 code. So, users with legacy code, or who choose to use only older language constructs, will still get the benefit of updated and actively maintained compilers.

As we, the authors of *Numerical Recipes*, watched the gestation and birth of Fortran 90 by its governing standards committee (an interesting process described by a leading Committee member, Michael Metcalf, in the Foreword to our Volume 2), it became clear to us that the right moment for moving Numerical Recipes from Fortran 77 to Fortran 90 was sooner, rather than later.

On the other hand, it was equally clear that Fortran-77-style programming — no matter whether with Fortran 77 or Fortran 90 compilers — is, and will continue for a long time to be, the "mother tongue" of a large population of active scientists, engineers, and other users of numerical computation. This is not a user base that we would willingly or knowingly abandon.

The solution was immediately clear: a two-volume edition of the Fortran *Numerical Recipes* consisting of Volume 1 (this one, a corrected reprinting of the previous one-volume edition), now retitled *Numerical Recipes in Fortran 77*, and a completely new Volume 2, titled *Numerical Recipes in Fortran 90: The Art of* Parallel *Scientific Computing*. Volume 2 begins with three chapters (21, 22, and 23) that extend the narrative of the first volume to the new subjects of Fortran 90 language features, parallel programming methodology, and the implementation of certain useful utility functions in Fortran 90. Then, in exact correspondence with Volume 1's Chapters 1–20, are new chapters B1–B20, devoted principally to the listing and explanation of new Fortran 90 routines. With a few exceptions, each Fortran 77 routine in Volume 1 has a corresponding new Fortran 90 version in Volume 2. (The exceptions are a few new capabilities, notably in random number generation and in multigrid PDE solvers, that are unique to Volume 2's Fortran 90.) Otherwise, there is no duplication between the volumes. The detailed explanation of the algorithms in this Volume 1 is intended to apply to, and be essential for, both volumes.

In other words: **You can use this Volume 1 without having Volume 2, but you can't use Volume 2 without Volume 1.** We think that there is much to be gained by having and using *both* volumes: Fortran 90's parallel language constructions are not only useful for present and future multiprocessor machines; they also allow for the elegant and concise formulation of many algorithms on ordinary single-processor computers. We think that essentially *all* Fortran programmers will want gradually to migrate into Fortran 90 and into a mode of "thinking parallel." We have written Volume 2 specifically to help with this important transition.

Volume 2's discussion of parallel programming is focused on those issues of direct relevance to the Fortran 90 programmer. Some more general aspects of parallel programming, such as communication costs, synchronization of multiple processers, etc., are touched on only briefly. We provide references to the extensive literature on these more specialized topics.

A special note to C programmers: Right now, there is no effort at producing a parallel version of C that is comparable to Fortran 90 in maturity, acceptance, and stability. We think, therefore, that C programmers will be well served by using Volume 2, either in conjuction with this Volume 1 or else in conjunction with the sister volume *Numerical Recipes in C: The Art of Scientific Computing*, for an educational excursion into Fortran 90, its parallel programming constructions, and the numerical algorithms that capitalize on them. C and C++ programming have not been far from our minds as we have written this two-volume version. We think you will find that time spent in absorbing the principal lessons of Volume 2's Chapters 21–23 will be amply repaid in the future, as C and C++ eventually develop standard parallel extensions.

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 Loredo, Matthew Choptuik, Gregory Cook, L. Samuel Finn, P. Deuflhard, Harold Lewis, Peter Weinberger, David Syer, Richard Ferch, Steven Ebstein, 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 influenced many of the routines in this book, and wrote or rewrote many more routines in its C-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.

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, TEX, Perl, Adobe Illustrator, and PostScript. Also used were a variety of FORTRAN 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 a large majority 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 (Chaper 3), integration (Chaper 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 `FORTRAN`-77. Versions of the book in `C`, `Pascal`, and `BASIC` are available separately. We have more to say about the `FORTRAN` 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 TEX, 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 our Web site (`http://www.nr.com`). They are packaged as a password-protected file, and you'll need to purchase a license to unpack them. You can get a single-screen license and password immediately, on-line, from the On-Line Store, with fees ranging from \$50 (PC, Macintosh, educational institutions' UNIX) to \$140 (general UNIX). Downloading the packaged software from the On-Line Store is also the way to start if you want to acquire a more general (multiscreen, site, or corporate) license.

- You can purchase media containing the programs from Cambridge University Press. Diskette versions are available in IBM-compatible format for machines running Windows 3.1, 95, or NT. CDROM versions in ISO-9660 format for PC, Macintosh, and UNIX systems are also available; these include both Fortran and C versions (as well as versions in Pascal and BASIC from the first edition) on a single CDROM. Diskettes purchased from Cambridge University Press include a single-screen license for PC or Macintosh only. The CDROM is available with a single-screen license for PC or Macintosh (order ISBN 0 521 576083), or (at a slightly higher price) with a single-screen license for UNIX workstations (order ISBN 0 521 576075). 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 trade@cup.cam.ac.uk (rest of world). Or, visit the Web sites `http://www.cup.org` (North America) or `http://www.cup.cam.ac.uk` (rest of world).

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 practicably 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

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	`sprsax`	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	`qrupdt`	update a QR decomposition
2.10	`rotate`	Jacobi rotation used by `qrupdt`
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

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`	extirpolate 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	`quadvl`	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	`fit`	least-squares fit data to a straight line