

The *Mathematica*[®] Journal

Volume 9, Issue 4

Contents

Departments

Staff and Contributors	666
Calendar	667
News Bulletins	668
New Products	670
New Resources	677
New Publications	678
Article Submissions to <i>The Mathematica Journal</i>	878
Classifieds	882

Columns

Tricks of the Trade <i>edited by Paul Abbott</i>	684
In and Out <i>edited by Paul Abbott</i>	699
Trott's Corner <i>by Michael Trott</i>	713

Feature Articles

Moment-Based Density Approximants <i>by Serge B. Provost</i>	727
Stochastic Integrals and Their Expectations <i>by Wilfrid S. Kendall</i>	757
Bootstrap Tutorial <i>by Hal Varian</i>	768
Stable Distributions in <i>Mathematica</i> <i>by Robert H. Rimmer and John P. Nolan</i>	776
Computational Order Statistics <i>by Colin Rose and Murray D. Smith</i>	790
Applications of Generating Functions in Nonparametric Tests <i>by Peter Weiß</i>	803
On the Numerical Accuracy of <i>Mathematica</i> 5.0 for Doing Linear and Nonlinear Regression <i>by Marc Nerlove</i>	824
Skew Densities and Ensemble Inference for Financial Economics <i>by H. D. Vinod</i>	852
Neighbourhoods of Independence for Random Processes via Information Geometry <i>by Khadiga Arwini and C. T. J. Dodson</i>	863

The electronic version of this issue is available at www.mathematica-journal.com/issue/v9i4.

The online edition of *The Mathematica Journal* includes downloadable versions of all articles and packages in this issue.

On the Cover

A patch of Penrose tiling created using the generalized dual method. See library.wolfram.com/infocenter/MathSource/1197.

Staff and Contributors

Managing Editor: George Beck

Project Manager: Peggy Pope

Design Team: Jeremy Davis, Megan Gillette

Compositor: Glenn Scholebo

Archives and Web Production: Jean Buck

Web Development: Andy Hunt, Lisa Denlinger

Editorial Staff: George Beck, Marcia Krause, Larry Adelston, Lorri Coey

Columnists: Paul Abbott, Michael Trott

Contributors: Khadiga Arwini, C. T. J. Dodson, Wilfrid S. Kendall, Marc Nerlove, John P. Nolan, Serge B. Provost, Robert H. Rimmer, Colin Rose, Murray D. Smith, Hal Varian, H. D. Vinod, Peter Weiß

Editorial Advisory Board: Steve Christensen (MathTensor, Inc.; Christensen and Associates, Inc.); Donald Glaser (University of California, Berkeley); Donald Knuth (Stanford University); Wolfram Koepf (University of Kassel); Silvio Levy (Mathematical Sciences Research Institute); Daniel Lichtblau (Wolfram Research, Inc.); David Lovelock (University of Arizona); Jerry Marsden (Caltech); Robert Moore (University of Washington, Seattle); Flip Phillips (Skidmore College); Bill Press (Harvard University); Sam Savage (University of Chicago); Dana Scott (Carnegie Mellon University); Lawrence M. Seiford (University of Michigan, Ann Arbor); Lynn Steen (St. Olaf College); Francis Sullivan (Institute for Defense Analyses); Bill Thurston (University of California, Davis); Michael Trott (Wolfram Research); Alfred van der Poorten (Macquarie University); Hal Varian (University of California, Berkeley); Eric Weisstein (Wolfram Research); Rob Wolff (Apple Computer, Inc.); and Stephen Wolfram (Wolfram Research).

Customer Services

www.mathematica-journal.com

+1-877-294-8184 (U.S. and Canada)

+1-217-398-6000

fax: +1-217-398-1108

subscriptions: subscribe@mathematica-journal.com

Advertising

The Mathematica Journal accepts advertising for all *Mathematica*-related products and services. Classified advertising is provided for *Mathematica* users. Rates are available at www.mathematica-journal.com/advertising. Contact the advertising department through the telephone numbers above, or send email to ads@mathematica-journal.com.

The Mathematica Journal is typeset in *Mathematica 5* using Janson, Futura, Math CMTT and the standard *Mathematica* fonts.

Publishing

The Mathematica Journal (ISSN 1047-5974) is published by Wolfram Media, Inc. Four-issue subscription rates in U.S. dollars: Electronic version: individuals, \$35; institutions, \$70; students, \$20. Electronic and print versions: individuals, \$75; institutions, \$200; students, \$50. Canadian/Mexican print subscriptions, add \$10. Print subscriptions outside North America, add \$32. Single print copy and back issue sales price: \$25. To subscribe or renew, call +1-217-398-6000 or +1-877-294-8184 (U.S. only), or visit our website at www.mathematica-journal.com to enter your subscription information. Postmaster: Send address changes to: *The Mathematica Journal*, P.O. Box 6359, Champaign, IL 61826-6359.

© 2005 Wolfram Media, Inc. No part of this publication may be reprinted or otherwise reproduced without the express permission of its publisher. No copyright claimed for U.S. government materials.

Mathematica, *MathLink*, and *MathSource* are registered trademarks of Wolfram Research, Inc. All other trademarks are the property of their respective owners. *Mathematica* is not associated with Mathematica Policy Research, Inc. or MathTech, Inc. Material in this publication may not be reproduced in any form without permission of the publisher. The opinions expressed in the *Journal* are the authors' own and do not necessarily reflect those of the authors' employers, the editors, or the publishers of the *Journal*.

Calendar

For the complete Wolfram Research calendar, see
www.wolfram.com/news/events.

Send your event notices to calendar@mathematica-journal.com.

2005 Highlighted Events

August

International *Mathematica* Symposium 2005

IMS 2005 Organizing Committee

August 5–8, The University of Western Australia, Perth, Western Australia

Info: internationalmathematicasymposium.org/IMS2005

September

Wolfram Research Tour of Russia and Ukraine

Wolfram Research

September 19–26, Novosibirsk, Moscow, St. Petersburg, and Kiev

Info: www.wolfram.com/services/seminars/russia2005

October

Wolfram Technology Conference 2005

Wolfram Research

October 6–8, Champaign, Illinois

Info: www.wolfram.com/news/events/techconf2005

News Bulletins

View more *Mathematica* news at www.wolfram.com/news.

***Mathematica* Ported to Intel Mac, Takes Center Stage at WWDC**

Mathematica is the first external application to run seamlessly on the forthcoming Intel Mac platform. Theodore Gray, director of user interface technology at Wolfram Research, made the announcement at Apple's Worldwide Developers Conference (WWDC) 2005, sharing the keynote address with Apple CEO Steve Jobs. For more information, go to www.wolfram.com/news/wwdc.html.

Wolfram *Publicon* Gives New Momentum to Open Access Publishing

Wolfram Research and BioMed Central have partnered to streamline Open Access publishing with Wolfram *Publicon*. *Publicon* simplifies the process of Open Access publishing—which provides free online access to research, while authors pay a small fee to cover costs—so submissions go through with minimal intervention. For more information, go to www.wolfram.com/news/bmc.html.

Wolfram Technology Conference 2005 Call for Submissions

Proposals for the Wolfram Technology Conference 2005 are being accepted now through August 1. This year's conference will cover a wide range of topics, with special sessions held to highlight how users have applied *Mathematica* to their work and have built custom solutions based on Wolfram technologies. Users of all levels and backgrounds, including students, are encouraged to submit. For more information, go to www.wolfram.com/news/events/techconf2005/submissions.html.

Wolfram Research Opens New Office in Italy

Wolfram Research Italia is now in full operation as the new face of Wolfram Research in Italy. The office provides local expertise with *Mathematica* technologies and more immediate access to customer support. For more information, go to www.wolfram.com/news/italia.html.

Ten Years in Asia

Wolfram Research Asia Ltd. recently celebrated the tenth anniversary of its Japan-based offices, marking a new milestone in the continued success of the direct sales and marketing liaison for Wolfram Research. This strong, personal relationship between Wolfram Research, its local resellers, and the sophisticated *Mathematica* user community has fostered impressive long-term stability in the company's large Asian markets. For more information, go to www.wolfram.com/news/wral10yrs.html.

New Products

Mathematica 5.2

Breaking the Memory Barrier with 64-Bit Computing, Adding Multicore Performance

Hot on the heels of *Mathematica* 5.1, itself released just eight months ago, 5.2 brings 64-bit technology to all supported platforms—an industry first. More than 4.3GB of memory (the 32-bit address limit) can now be addressed, and high-precision or large numbers are processed in 64-bit rather than 32-bit digit chunks for faster computation.

“*Mathematica* users push the limits of computing—constantly requiring more memory and computational horsepower. That is why we have decided to be first to market with all-platform 64-bit and multicore support,” said Roger Germundsson, director of research and development. “And we’ve been able to do this because of our state-of-the-art software engineering processes and close working partnerships with platform vendors such as Microsoft, Apple, Intel, and Sun Microsystems.”

“*Mathematica* is now the ideal environment for large computations and simulations,” said Tom Wickham-Jones, director of kernel technology. “From 5.0 onwards we’ve dramatically sped up computation, reduced memory usage, and introduced grid computing; now with 5.2 we’ve enabled *Mathematica* to take advantage of more memory and more CPUs for computation, too.”

Mathematica 5.2 also supports automatically threaded numerical linear algebra on all mainstream platforms, enabling linear algebra operations to automatically run in parallel on all available processor cores whether multiple or multicore CPUs. Mainstream multicore-based systems are now available and are expected to be ubiquitous by early next year, including in notebooks and entry-level systems.

Mathematica 5.2 contains many additional enhancements, including:

- 64-bit-enhanced arbitrary-precision numerics
- Vector-based performance enhancements
- Automatic binary installation selection
- Bundled notebook indexing for desktop search
- SSH support for secure remote kernels
- vCard and RSS import
- New algorithms for symbolic differential equations
- Enhanced performance for linear Diophantine systems

- Enhanced quadratic quantifier elimination
- Singular-case support for high-level special functions
- Enhanced statistics charts
- *MathematicaMark* 5.2 benchmark now covering grids and clusters

“If you’re at the frontiers of technical work, you need computing technology that is as well,” said Conrad Wolfram, director of strategic development. “*Mathematica* 5.2 again demonstrates our commitment to rapid and continuing support for the latest computing technology.”

More info

www.wolfram.com/products/mathematica/newin52

Wolfram *Mathematica CalcCenter* 3

The New, Easy-to-Use “Lite” Version of Mathematica

Wolfram Research announces *Mathematica CalcCenter* 3 software—characterized as having “the power of *Mathematica*, the ease of a calculator.” Ideal for the engineer, scientist, educator, or student, *Mathematica CalcCenter* 3 offers an award-winning interface—getting users up and running in as little as 10 minutes—while delivering industrial-strength performance and trustworthy results.

Mathematica CalcCenter 3 has been derived from *CalculationCenter*, but is now upgraded to meet the standards of the *Mathematica* 5 product family.

“Think of *Mathematica CalcCenter* 3 as *Mathematica* lite—with added ease of use,” said Jon McLoone, lead developer of *Mathematica CalcCenter* 3. “We’ll be focused on migrating Mathcad, Maple, and technical Excel users fazed by the full *Mathematica* but wanting its trustworthy results and industrial-strength performance.”

Because *Mathematica CalcCenter* 3 utilizes the base technology of the latest version of *Mathematica*, there is a straightforward upgrade path for those who subsequently need *Mathematica* Professional’s unsurpassed power.

“When you run out of steam with Excel, Maple, or Mathcad, where do you go? If you ever need more power than *Mathematica CalcCenter*, you can transition smoothly to *Mathematica*,” said McLoone. “Moreover, workgroups can use *Mathematica* and *Mathematica CalcCenter* interchangeably, swapping notebooks as required.”

Mathematica CalcCenter 3 enjoys other benefits by sharing common technology with *Mathematica*. “Because *Mathematica CalcCenter* 3 is an OEM product of *Mathematica*, it piggybacks on *Mathematica*’s development rather than having to be built from scratch in its own right,” said Conrad Wolfram, director of strategic development. “That’s what allows us to deliver better price-to-performance than any competitors.”

In comparison to *CalculationCenter 2*, *Mathematica CalcCenter 3* adds a host of new features, including:

- Data analysis now up to 70 times faster—matches *Mathematica 5*'s record-breaking speed
- New-generation optimized numerical solvers for ordinary and partial differential equations
- Fully optimized support for Mac OS X
- Support for computations with exact fractions
- Support for vector and array functions in numeric solvers
- Flexible import and export of DICOM, PNG, SVG, STDS, DXF, FITS, STL, and sparse matrix formats Matrix Market and Harwell-Boeing
- Improved export of XHTML, now generating cascading style sheets
- Extended Copy and Paste on Windows for using typeset formulas in other applications
- Streamlined design and user interface improvements
- Over 50 improvements to quality and robustness

Mathematica CalcCenter 3 is available for Windows and Mac OS X.

More info

www.wolfram.com/products/calccenter

gridMathematica 2

A Major Enhancement to the Premier Software Environment for Supercomputing

gridMathematica 2 is now available from Wolfram Research. This new version of the widely used parallel computing software harnesses all the capabilities of *Mathematica 5.1* and is optimized for virtually all Top 500 supercomputers, heterogeneous grids or clusters, and personal workstations. With gridMathematica, scientists and engineers can easily develop powerful parallel applications on a PC and deploy them seamlessly on any grid or supercomputer.

Features of gridMathematica 2 include significant scheduling and performance enhancements, automatic parallelization of wider classes of computations, programmable scheduling for problem-specific tuning, support for speculative parallelism, seamless execution on scalar and parallel setup, improved failure recovery and user configurability, universal database connectivity, web services support, high-speed binary I/O, industrial-strength string manipulation, and full optimization for all 64-bit architectures that *Mathematica* supports.

Building on the rich foundation of *Mathematica* 5.1 and *Parallel Computing Toolkit* 2, *gridMathematica* is perfectly suited for numerical supercomputing applications as well as parallel processing of complex non-numeric data such as images, 2D and 3D vector graphics, geospatial information, genomic sequences, algebraic formulas, logical and program constructs, and general symbolic data.

Since its introduction in 2002, *gridMathematica* has been successfully deployed by many major corporations, government organizations, and universities, including Aerospace Corp, Argonne National Laboratory, Columbia University, Dow Chemical, Istituto Nazionale di Fisica Nucleare, Kyoto University, Los Alamos National Laboratory, MIT, NASA Langley, Queen's University, RAND Corporation, St. Jude Children's Research Hospital, Seagate Technology, Silicon Graphics, Space Telescope Science Institute, Thomson Multimedia, University of California Berkeley, University of North Carolina, University of Tokyo, Yale University, and many more.

"*gridMathematica* is a powerful tool for developing parallel applications for the Windows platform," said Dennis Oldroyd, director of the Windows Server Division at Microsoft Corporation. "Now customers can use a standard PC to write large parallel algorithms and then deploy them on larger clusters and grids with limited porting time. This tool helps simplify the development and deployment of computationally intensive parallel applications on the Windows platform."

gridMathematica 2 is available for all common Linux and Unix systems, Windows, and Mac OS X. Computers in the cluster do not have to run the same operating system. Introductory training on *gridMathematica* is offered by the Wolfram Education Group.

More info

www.wolfram.com/products/gridmathematica

webMathematica 2.2

Upgraded with New Mathematica Features

The new *webMathematica* 2.2 from Wolfram Research capitalizes on the strengths of *Mathematica* 5.1 to provide expanded web-development options. Building on *Mathematica* 5.1's many new capabilities, especially for working with large-scale, diverse types of data, *webMathematica* increases its advantage as the leading all-in-one solution for creating powerful interactive websites for research, business, and education.

There are more than 50 new functions, toolkits, and performance improvements in *Mathematica* 5.1. Highlights of special interest for use with *webMathematica* 2.2 include:

- Built-in universal database connectivity
- Industrial-strength string manipulation

- Integrated web services support
- Authoring system preview
- Highly optimized binary data I/O
- Support for new 64-bit architectures
- Additional graphic, web, and matrix import-export formats

With these features, *webMathematica* 2.2 opens the door to new kinds of application development on a wider range of platforms. Industrial-strength string manipulation makes *webMathematica* a good fit for data mining and bioinformatics applications, while support for web services makes it easy for users to harness financial, environmental, language translation, and other information resources. “There are endless possibilities with web services linked with a *webMathematica* interface,” said Chris Williamson, developer of web technologies at Wolfram Research.

DatabaseLink provides a ready-made solution for integrating *webMathematica* with any standard SQL database. This makes *webMathematica* an ideal web-based environment for analysis, computation, and reporting—which complements the storage and searching capabilities of databases. “There is increasing demand for web-accessible databases,” said Williamson. “*webMathematica* enables delivery of sophisticated, data-driven applications and analysis tools that make corporate data accessible at any time and from anywhere.”

webMathematica continues to answer the problem of how to create and distribute technical computing application solutions in today’s networked environment. Built on platform-independent standards such as HTML, Java, and Java Servlet technology, *webMathematica* 2.2 seamlessly integrates the advances of *Mathematica* 5.1 into virtually any infrastructure.

More info

www.wolfram.com/products/webmathematica

***Mathematica* Applications**

Parallel Computing Toolkit 2

Parallel Computing Toolkit 2 is a versatile parallel computing solution for anyone with access to more than one processor. This *Mathematica* application package enables users to tackle large-scale problems and build platform-independent parallel programs with their existing *Mathematica* installations.

More info

www.wolfram.com/products/applications/parallel

Analog Insydes 2.1

Analog Insydes is a *Mathematica* application package for the modeling, analysis, and design of analog electronic circuits, tailored specifically for industrial applications. This version adds new interfaces for Spectre, an extended device model library, and full integration of high-performance *MathLink* binaries on Mac OS X.

More info

www.wolfram.com/products/applications/insydes

MathCode F90

MathCode F90 is a powerful tool for developing highly optimized Fortran 90 applications directly in *Mathematica*. It provides a platform for the rapid development of production-quality code for heavy simulations and other expensive computations. The automatic code generation of *MathCode F90* is also a safeguard against the typing errors and associated bugs encountered when using conventional methods for prototyping and implementation.

More info

www.wolfram.com/products/applications/mathcodef90

MathOptimizer Professional

MathOptimizer Professional enables the global and local solution of a general class of continuous optimization problems, and now supports Linux and several new compilers. It combines the power of *Mathematica* with the established LGO (Lipschitz Global Optimizer) solver suite to offer sophisticated application development tools and solver functionality.

More info

www.wolfram.com/products/applications/mathoptpro

Operations Research 3.1

Operations Research works with *Mathematica* to solve problems in linear optimization, quadratic programming, shortest path, and combinatorial optimization and heuristics. It solves a broad range of typical problems encountered in university courses and in daily practical work, and can model and solve optimization problems in a very convenient and interactive manner.

More info

www.wolfram.com/products/applications/operationsresearch

UnRisk PRICING ENGINE 2.2

UnRisk PRICING ENGINE combines speed with precision in the analysis of financial derivatives. By integrating *Mathematica* with optimized C++ libraries, it provides an extremely reliable ready-built solution that offers both the flexibility of a high-level programming language and the speed necessary for production use.

More info

www.wolfram.com/products/applications/unrisk

Updates for *Mathematica* Compatibility

Many other *Mathematica* application packages have been updated for compatibility with the latest versions of *Mathematica*. These updates are free to registered users of the packages, and are ready for immediate download.

More info

www.wolfram.com/products/applications/updates

New Resources

MathWorld

Major enhancements have been made to *MathWorld*, the web's most extensive mathematics resource, which now features interactive web*Mathematica*-powered calculators and graphs, dynamic 3D visualization tools, colorful animations, and more. Complementing these additions to the main site, the *MathWorld* Classroom is also newly online, providing quick introductions to topics in algebra, geometry, trigonometry, statistics, precalculus, calculus, number theory, and all other high school math courses. Visit *MathWorld* at mathworld.wolfram.com.

Wolfram Notebook Indexer for Spotlight and Google Desktop Search

Wolfram *Notebook Indexer* is now available for Mac OS X 10.4 Tiger's Spotlight and as a free public beta for Google Desktop Search. This new plug-in, included with *Mathematica* 5.2, lets users perform text searches of all Wolfram notebook (.nb) documents on their local hard drives, including those that are a part of *Mathematica*'s online help system. Download the Google Desktop Search plug-in at library.wolfram.com/infocenter/Utilities/5596 or the Spotlight plug-in at library.wolfram.com/infocenter/Utilities/5614.

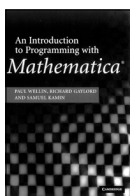
Wolfram Teacher Network

High school teachers in math and science can sign up now to join the Wolfram Teacher Network, a newly expanded free educational resource built around the guidance of *Mathematica* mentors. The program offers experienced and expert advice to teachers who want to push the limits of traditional classroom learning with *Mathematica*. The Wolfram Teacher Network is also looking for more *Mathematica* mentors, and all teachers who share an enthusiasm for *Mathematica* are encouraged to apply. For more information, go to www.wolfram.com/precollege/network.

Wiki-webMathematica

Luc Barthelet, senior vice president at Electronic Arts and general manager for Maxis during the development of The Sims video games, has launched Wiki-web-*Mathematica*, a new online *Mathematica* user forum that utilizes the power of web*Mathematica* with the freedom of the wiki user interface. It supplements MathGroup (forums.wolfram.com/mathgroup) by allowing *Mathematica* users to add and edit content—including their notebooks, which are properly displayed online. Visit Wiki-web*Mathematica* at www.mathematica-users.org.

New Publications



Introduction to Programming with *Mathematica*, Third Edition

Paul R. Wellin, Richard J. Gaylord, and Samuel N. Kamin, 2005,
Cambridge University Press, 550 pp., hardcover

This book introduces the *Mathematica* programming language to a wide audience. Since the last edition of this book was published, significant changes have occurred in *Mathematica* and its use worldwide. Keeping pace with these changes, this substantially larger, updated text includes new and completely revised chapters on numerics and on procedural, rule-based, and front end programming, and gives significant coverage to the latest features and functions in *Mathematica* 5.

Contents

An Introduction to *Mathematica* | The *Mathematica* Language | Lists | Functional Programming | Procedural Programming | Rule-Based Programming | Recursion | Numerics | Graphics Programming | Front End Programming | Examples and Applications | Writing Packages | Appendix A: How Expressions Are Evaluated | Appendix B: Debugging | References | Solutions to Exercises

Available:

store.wolfram.com/view/book/ISBN0521846781.str

Book Review

Bruce Torrence

Associate Professor and Chair
Department of Mathematics
Randolph-Macon College
Ashland, VA 23005

Third editions can be a blasé affair. They rarely garner the attention of a new release, and in many cases they offer little more than minimal revisions and essential updates. In the case of programming books, you might expect the changes to primarily reflect new developments in the underlying software. That is not the case here. The third edition of *An Introduction to Programming with Mathematica* (hereafter IPM3) is in many respects an entirely new work. It has

been revised from cover to cover, with many organizational changes and a substantial amount of new material. It includes coverage up to and including *Mathematica* 5.1. While the second edition was very well received, the third edition strikes me as an essential document.

The organization of this edition has changed substantially. The first two chapters have been renamed and rewritten but still provide an introduction to the software and the syntax and structure of the *Mathematica* programming language. I was struck immediately by the lucidity of the prose and the order of the topics. While the interweaving of concepts makes any linear ordering of *Mathematica* topics necessarily deficient, the choices made in IPM3 work well. The text also benefits from the considerable experience that Paul Wellin, the primary author for the third edition, has accrued in his role as director of the Wolfram Education Group. It will be obvious to any educator that a great amount of reflection and experience in teaching these materials has informed and refined their presentation since the second edition.

The third chapter, Lists, retains much of its original character and adds a new section on strings. But from here on out it is in many ways a completely new book. Chapters four through six are organized by three distinct programming styles: functional, procedural, and rule-based programming, respectively. Much of this material appeared in the second edition, and in much the same order, but the organization obscured this interpretation. However, much here is also new. For example, `Block` and `With` are now discussed (they did not appear in the second edition) alongside `Module`. The treatment is not meant to be comprehensive, but it is sufficient to quickly see the major distinctions between these three commands, and it is a clear improvement over the second edition. Additionally, `Thread` appeared only in the exercises in the second edition. It is discussed in greater detail in this edition, with carefully chosen examples, such as threading `Rule` over two lists to get a list of replacement rules, a simple and useful technique.

Moreover, the third edition makes excellent use of simple commands that reveal and emphasize the underlying structure of expressions, such as following an outputted list with the command `Map[FullForm,%]`. The liberal inclusion of such commands encourages the reader to think about the underlying structure of expressions throughout, and this lends considerably to the book's merit.

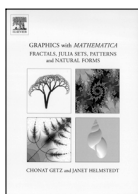
Chapters on Recursion, Numerics, and Graphics Programming are preserved in IPM3, with a nice section on merge sorting added to the Recursion chapter. The Numerics chapter is largely rewritten. In particular, the section on precision and accuracy is greatly expanded and provides as clear an introduction to those ideas as I have seen anywhere. New sections on sparse arrays and packed arrays are also included to provide coverage up to Version 5.1. In the Graphics chapter, the material is reorganized, but the basic content is preserved.

A new chapter on Front End Programming is a welcome addition. After carefully introducing the idea that cells and notebooks are themselves expressions, a brief but valuable presentation on such concepts as `TextData`, `BoxData`, and `Graphics`Data` is given. `GridBoxes` are presented in some detail (there is a gem of a short program included to produce truth tables for logical expressions), and there is a

brief but informative section on button programming. New also are two appendices, covering how expressions are evaluated and debugging.

It is worthy of mention that unlike the second edition (which was written in $\text{\LaTeX}$) the third edition is produced directly from *Mathematica* 5 notebooks, using style sheets made by Wellin and slightly modified commands from the standard *AuthorTools* package to produce the table of contents, index, and so on. That is, anyone could, in theory, follow a similar path and use *Mathematica* to produce a production quality text. This is not the first book to be written in *Mathematica*, but it is far and away the most beautiful. From page layout to production quality, the work is on par with any technical text produced by any publisher. This should give future authors of *Mathematica* texts something to think about.

In conclusion, IPM3 is a welcome addition to the literature. The choice of topics and their presentation are lucidly informed by considerable experience. The writing style is straightforward and clear. The authors stick resolutely to their goal of producing an introductory text, not a complete treatment, on programming in *Mathematica*. As such it is highly successful, full of useful examples that show how the ideas can be immediately and fruitfully exploited. The only bad news is that unlike many other third editions, owners of IPM2 may find that they need to shell out for the new edition.



Graphics with *Mathematica*: Fractals, Julia Sets, Patterns and Natural Forms

Chonlat Getz and Janet Helmstedt, 2004, Elsevier, 322 pp., hardcover, includes CD-ROM

This book gives detailed instructions for constructing interesting 2D and 3D fractals using iterated functions systems in *Mathematica*. The programs used to generate these graphics are easily adaptable to many variations, and a range of different coloring techniques are introduced. Many different types of Julia sets and parameter sets are constructed, such as the Mandelbrot set. The reader interested in fractals will obtain practice in graphics and coloring in the earlier chapters “Using Color in Graphics” and “Patterns Constructed from Straight Lines.”

Highlights include:

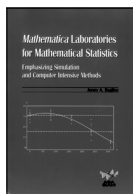
- Brilliant graphic images
- Interesting introduction to *Mathematica* for beginners
- Easy constructions
- A variety of coloring techniques
- Programs easily adaptable to many variations
- Constructions useful for courses in discrete dynamics and fractals

Contents

Basics | Using Color in Graphics | Patterns Constructed from Straight Lines | Orbits of Points under a $C \rightarrow C$ Mapping | Using Roman Maeder's Packages Affine Maps, Iterated Function Systems and Chaos Game to Construct Affine Fractals | Constructing Non-affine and 3D Fractals using the Deterministic and Random Algorithms | Julia and Mandelbrot Sets Constructed Using the Escape-Time Algorithm and Boundary Scanning Method | Miscellaneous Design Ideas | Appendices | Bibliography | Index | Colour Figures

Available:

store.wolfram.com/view/book/ISBN044451760X.str



Mathematica Laboratories for Mathematical Statistics: Emphasizing Simulation and Computer Intensive Methods

Jenny A. Baglivo, 2005, SIAM-ASA, 260 pp., paperback, includes CD-ROM

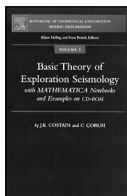
This book presents a concise introduction to the concepts of probability theory and mathematical statistics. The accompanying in-class and take-home computer laboratory activities reinforce the techniques introduced in the text and are accessible to students with little or no experience with *Mathematica*. These laboratory materials present applications in a variety of real-world settings, with data from epidemiology, environmental sciences, medicine, social sciences, physical sciences, manufacturing, engineering, marketing, and sports. Prerequisites include multivariable calculus and familiarity with the basics of set theory, vectors, matrices, and problem-solving using a computer.

Contents

Preface | Introductory Probability Concepts | Discrete Probability Distributions | Continuous Probability Distributions | Mathematical Expectation | Limit Theorems | Transition to Statistics | Estimation Theory | Hypothesis Testing Theory | Order Statistics and Quantiles | Two Sample Analysis | Permutation Analysis | Bootstrap Analysis | Multiple Sample Analysis | Linear Least Squares Analysis | Contingency Table Analysis

Available:

store.wolfram.com/view/book/ISBN0898715660.str



Basic Theory of Exploration Seismology [with *Mathematica*]

John K. Costain and Cahit Çoruh, 2004, Elsevier, 571 pp., hardcover, includes CD-ROM

This volume provides the basic theory and principles behind imaging the subsurface of the Earth using reflection and refraction seismology, how to improve these images via data processing, and how to understand reflection seismology data sets. It is of particular interest for geologist and geophysicist students and professionals.

The authors draw on their own copious experience and data to provide real-world results and illustrations of seismic images. The accompanying CD-ROM includes complete *Mathematica* programs, movies, and plot statements to let readers explore and extend the examples.

Contents

Introduction | Groundwork | Fourier Transforms | Computational Considerations | Synthetics and Velocity Functions | Traveltime Curves and Velocity | Seismic Source Wavelets | Wavelet Shaping and Deconvolution

Available:

store.wolfram.com/view/book/ISBN0080370195.str



Applied Laplace Transforms and z-Transforms for Scientists and Engineers: A Computational Approach using a *Mathematica* Package

Urs E. Graf, 2004, Birkhäuser Verlag, 500 pp., hardcover, includes CD-ROM

This book presents theory and applications of Laplace and z-transforms together with a *Mathematica* package developed by the author, which includes algorithms for the numerical inversion of Laplace transforms. This allows the symbolic computation capability of *Mathematica* to be used in favor of the Laplace and z-transformations, making them more accessible to engineers and scientists.

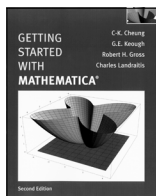
The emphasis is on the computational and applied sides, particularly for control engineering, electrical engineering, and mechanics (heat conduction, diffusion, vibrations). Many worked-out examples from applied sciences and engineering illustrate the theory's applicability and how to use the package.

Contents

Laplace Transformation | z-Transformation | Laplace Transforms with the Package | z-Transformation with the Package | Applications to Automatic Control | Laplace Transformation: Further Topics | z-Transformation: Further Topics | Examples from Electricity | Examples from Control Engineering | Heat Conduction and Vibration Problems | Further Techniques | Numerical Inversion of Laplace Transforms | Appendix: Package Commands

Available:

store.wolfram.com/view/book/ISBN3764324279.str



Getting Started with *Mathematica*, Second Edition

C-K. Cheung, G. E. Keough, Robert H. Gross, and Charles Landraitis, 2005, John Wiley & Sons, 231 pp., paperback

This is a step-by-step tutorial with an emphasis on getting the reader using *Mathematica* as quickly as possible. It can be used as a reference for students in any mathematics or science course that requires *Mathematica*, or as a learning tool for any new *Mathematica* users.

Contents

Basic *Mathematica* Commands | 2-D Pictures in *Mathematica* | *Mathematica* for One-Variable Calculus | 3-D Pictures in *Mathematica* | *Mathematica* for Advanced Calculus | Using *Mathematica* in Statistics | Advanced Features of *Mathematica* | Appendices

Available:

store.wolfram.com/view/book/ISBN0471478156.str

Tricks of the Trade

Edited by Paul Abbott

This is a column of programming tricks and techniques, most of which, we hope, will be contributed by our readers, either directly as submissions to *The Mathematica Journal* or as an edited answer to a question posted in the *Mathematica* newsgroup, comp.soft-sys.math.mathematica.

■ Interpolation with Noise

Vittorio G. Caffa

caffa@iabg.de

In “Interpolation,” *TMJ*, 9(2), 2004, pp. 306–310, a method was presented for computing derivatives of a sampled function. This method only works if the data is unaffected by measuring error. If models of the data source and the measurement noise are available, you can use (for example) Kalman filtering techniques for computing an optimal estimation (see mathworld.wolfram.com/KalmanFilter.html). Alternatively, you can achieve good results using a simple filtering technique.

Consider again the exact function $y(x) = x \sin(x) - x/2$.

```
In[1]:= y(x_) := x Sin[x] - x/2;
```

Take x from a uniform random distribution over the interval $(a, b) = (1, 12)$, and append the endpoints.

```
In[2]:= {a, b} = {1, 12};  
SeedRandom[4321];  
xs = Join[{a}, Table[Random[Real, {a, b}], {25}], {b}];
```

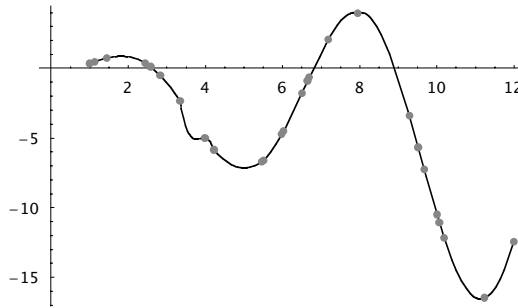
Now sample $y(x)$ over the interval adding Gaussian noise corresponding to measurement error.

```
In[3]:= Needs["Statistics`ContinuousDistributions`"];  
f(x_) := {x, y(x) + Random[NormalDistribution[0, 0.02]]};  
data = f /@ xs;
```

Superimpose the sampled data over a plot of its interpolated function to visualize the $\{x, y\}$ data. It is clear that the interpolation is not smooth, so computing derivatives will be problematic.

```
In[6]:= y_int = Interpolation[data];
```

```
In[7]:= Plot[yint(x), {x, a, b}, Epilog -> {Hue[1], AbsolutePointSize[3], Point /@ data}]
```



A simple filtering technique involves computing a truncated Fourier series of the function $y(x)$, that is,

$$y_n(x) = a_0 + \sum_{k=1}^n (a_k \cos(k \omega x) + b_k \sin(k \omega x)).$$

Requiring the wavelength λ to satisfy $\lambda = 2\pi/\omega > 2(b-a)$ avoids wrap-around effects. As only the frequencies up to $n\omega$ are considered, the method acts like a low-pass filter. The value of the parameter n should be tuned to obtain the best results. This approach should be compared with that used in “Spectral Analysis of Irregularly Sampled Data,” *TMJ*, 7(1), 1997, pp. 27–28.

Here is the Fourier basis set for arbitrary n and λ .

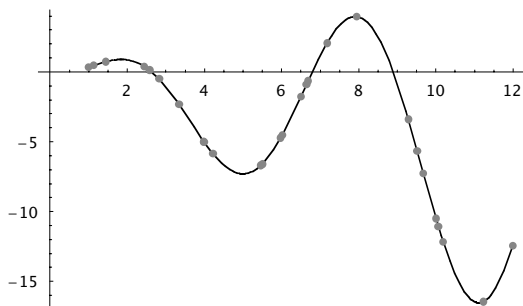
```
In[8]:= basis[n_, λ_] :=  
  With[{ω = 2 π / (N[λ]), k = Range[n]}, Join[{1}, cos(k ω x), sin(k ω x)]]
```

The best truncated Fourier expansion, y_n , for $\lambda = 30 > 2(b-a) = 22$ and $n = 5$ is computed using **Fit**.

```
In[9]:= y5 = Function[x, Evaluate[Fit[data, basis[5, 30], x]]]  
Out[9]= Function[x, 69.3068 cos(0.20944 x) + 95.0578 cos(0.418879 x) -  
  45.9663 cos(0.628319 x) + 5.88829 cos(0.837758 x) - 5.53855 cos(1.0472 x) +  
  186.779 sin(0.20944 x) - 79.2066 sin(0.418879 x) - 23.3035 sin(0.628319 x) +  
  8.88347 sin(0.837758 x) + 1.3561 sin(1.0472 x) - 118.99]
```

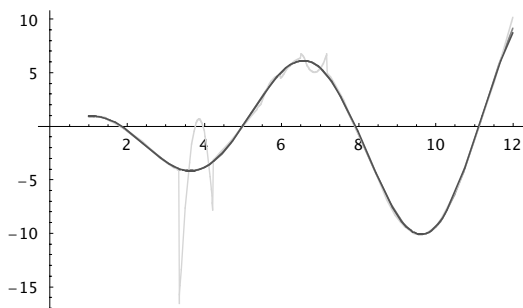
Here is the sampled data superimposed over a plot of y_5 .

```
In[10]:= Plot[y5(x), {x, a, b}, Epilog → {Hue[1], AbsolutePointSize[3], Point /@ data}]
```



Compare the first derivatives; the red curve denotes the exact value (—), the green curve denotes the interpolated function (—), and the blue curve denotes the truncated Fourier fit y_5 (—). The agreement between the exact function and the truncated Fourier fit is excellent.

```
In[11]:= Plot[{y'(x), y'_int(x), y'_5(x)}, {x, a, b}, PlotStyle → {Hue[0], Hue[1/3], Hue[2/3]}]
```



■ Using Reduce To Solve The Kuhn–Tucker Equations

Frank J. Kampas

fkampas@msn.com

The functions **Maximize** and **Minimize** give exact solutions to simple nonlinear optimization problems with equality and inequality constraints. Another approach is to set up Lagrange multipliers for equality constraints and the Kuhn–Tucker equations for inequality constraints (see mathworld.wolfram.com/Kuhn-TuckerTheorem.html) and solve using **Reduce**. In some cases, this approach can solve problems which cannot directly be solved with **Maximize** and **Minimize**. The following function demonstrates this approach.

```

In[1]:= KuhnTucker[obj_, cons_List, vars_, domain___) :=
Module[{consconvrule = {x_ ≥ y_ → y - x ≤ 0,
  x_ > y_ → y - x < 0, x_ = y_ → x - y = 0, x_ ≤ y_ → x - y ≤ 0},
  stdcons, eqcons, ineqcons, lambdas, mus, numequals,
  numinequals, lagrangian, eqs1, eqs2, eqs3},
stdcons = cons /. consconvrule; eqcons = Cases[stdcons, x_ = 0 → x];
ineqcons = Cases[stdcons, x_ ≤ 0 → x];
numequals = Length[eqcons]; numinequals = Length[ineqcons];
lambdas = Array[λ, numequals]; mus = Array[μ, numinequals];
lagrangian = obj + lambdas.eqcons + mus.ineqcons;
eqs1 =  $\left( \frac{\partial \text{lagrangian}}{\partial \#1} = 0 \ \& \right) /@ \text{vars}$ ; eqs2 = Thread[mus ≥ 0];
eqs3 = Table[mus[[i]] ineqcons[[i]] = 0, {i, numinequals}];
Reduce[Join[eqs1, eqs2, eqs3, cons],
Join[vars, lambdas, mus], domain, Backsubstitution → True]]

```

Here is an example with an equality constraint in which both the maximum and minimum are obtained. Defining the objective function with a constraint and listing it first in the list of variables places the objective function first in the results.

```

In[2]:= KuhnTucker[objf, {objf = x + y + z, x2 + y2 + z2 == 1}, {objf, x, y, z}]
Out[2]=  $\left\{ \text{objf} = -\sqrt{3} \wedge x = -\frac{1}{\sqrt{3}} \wedge y = -\frac{1}{\sqrt{3}} \wedge z = -\frac{1}{\sqrt{3}} \wedge \right.$ 
 $\left. \lambda(1) = -1 \wedge \lambda(2) = \frac{\sqrt{3}}{2} \right\} \vee \left\{ \text{objf} = \sqrt{3} \wedge x = \frac{1}{\sqrt{3}} \wedge \right.$ 
 $\left. y = \frac{1}{\sqrt{3}} \wedge z = \frac{1}{\sqrt{3}} \wedge \lambda(1) = -1 \wedge \lambda(2) = -\frac{\sqrt{3}}{2} \right\}$ 

```

Equivalent results are obtained by using **Maximize** and **Minimize**.

```

In[3]:= Maximize[objf, {objf = x + y + z, x2 + y2 + z2 == 1}, {objf, x, y, z}] // Simplify
Out[3]=  $\{\sqrt{3}, \{\text{objf} \rightarrow \sqrt{3}, x \rightarrow \frac{1}{\sqrt{3}}, y \rightarrow \frac{1}{\sqrt{3}}, z \rightarrow \frac{1}{\sqrt{3}}\}\}$ 

In[4]:= Minimize[objf, {objf = x + y + z, x2 + y2 + z2 == 1}, {objf, x, y, z}] // Simplify
Out[4]=  $\{-\sqrt{3}, \{\text{objf} \rightarrow -\sqrt{3}, x \rightarrow -\frac{1}{\sqrt{3}}, y \rightarrow -\frac{1}{\sqrt{3}}, z \rightarrow -\frac{1}{\sqrt{3}}\}\}$ 

```

KuhnTucker can solve the problem when the righthand side of the constraint is changed from a number to a symbol.

In[5]:= **KuhnTucker**[objf, {objf == x + y + z, x² + y² + z² == c}, {objf, x, y, z}]

$$\begin{aligned} \text{Out[5]} = & \left(\text{objf} = -\sqrt{3} \sqrt{c} \wedge x = -\frac{\sqrt{c}}{\sqrt{3}} \wedge y = -\frac{\sqrt{c}}{\sqrt{3}} \wedge \right. \\ & \left. z = -\frac{\sqrt{c}}{\sqrt{3}} \wedge \lambda(1) = -1 \wedge c \neq 0 \wedge \lambda(2) = \frac{\sqrt{3}}{2\sqrt{c}} \right) \vee \\ & \left(\text{objf} = \sqrt{3} \sqrt{c} \wedge x = \frac{\sqrt{c}}{\sqrt{3}} \wedge y = \frac{\sqrt{c}}{\sqrt{3}} \wedge z = \frac{\sqrt{c}}{\sqrt{3}} \wedge \right. \\ & \left. \lambda(1) = -1 \wedge c \neq 0 \wedge \lambda(2) = -\frac{\sqrt{3}}{2\sqrt{c}} \right) \end{aligned}$$

Maximize and **Minimize** give an error message in this situation.

In[6]:= **Maximize**[objf, {objf == x + y + z, x² + y² + z² == c}, {objf, x, y, z}]

Maximize::conv : The constraint x² + y² + z² == c contains a nonconstant expression c independent of variables {objf, x, y, z}. More...

Out[6]= **Maximize**[objf, {objf == x + y + z, x² + y² + z² == c}, {objf, x, y, z}]

In the following example, the constraint $x \leq b$ is only active if b is less than 1. Therefore, the solution obtained covers $b \leq 1$ and $b \geq 1$ separately.

In[7]:= **KuhnTucker**[objf, {objf == x² - 2 x, x ≤ b}, {objf, x}]

$$\begin{aligned} \text{Out[7]} = & (b \leq 1 \wedge \text{objf} = b^2 - 2b \wedge x = b \wedge \lambda(1) = -1 \wedge \mu(1) = -2(b-1)) \vee \\ & (b \geq 1 \wedge \text{objf} = -1 \wedge x = 1 \wedge \lambda(1) = -1 \wedge \mu(1) = 0) \end{aligned}$$

When the solution is degenerate, all the results can be obtained.

In[8]:= **KuhnTucker**[objf, {objf == x + y, x + y == 1}, {objf, x, y}]

$$\text{Out[8]} = \text{objf} = 1 \wedge y = 1 - x \wedge \lambda(1) = -1 \wedge \lambda(2) = -1$$

The following problem can be solved quite easily if the objective function is not defined as a constraint. Adding the extra constraint greatly increases the solution time. The problem cannot be solved by **Minimize** or **NMinimize** unless a change of variables is used to eliminate the square roots.

In[9]:= **KuhnTucker**[$\frac{1}{2}(t_b - \sqrt{q_b} + t_l - \sqrt{q_l})$, {t_l ≥ 20 q_l, t_b ≥ 2 q_b,
t_l - 20 q_l ≥ t_b - 20 q_b, t_b - 2 q_b ≥ t_l - 2 q_l}, {q_l, q_b, t_l, t_b}]

$$\begin{aligned} \text{Out[9]} = & q_l = \frac{1}{5776} \wedge q_b = \frac{1}{16} \wedge t_l = \frac{5}{1444} \wedge \\ & t_b = \frac{185}{1444} \wedge \mu(1) = 1 \wedge \mu(2) = 0 \wedge \mu(3) = 0 \wedge \mu(4) = \frac{1}{2} \end{aligned}$$

Using **Reduce** with domain $\mathbb{R}$ specified eliminates complex solution candidates.

```
In[10]:= KuhnTucker[objf, {objf == x + y, x^3 + y^3 == 1}, {objf, x, y}, Reals] //
FullSimplify
```

$$\text{Out[10]} = \text{objf} = 2^{2/3} \bigwedge x = \frac{1}{\sqrt[3]{2}} \bigwedge y = \frac{1}{\sqrt[3]{2}} \bigwedge \lambda(1) = -1 \bigwedge \sqrt[3]{2} \lambda(2) = -\frac{2}{3}$$

Note, however, that using Lagrange multipliers and the Kuhn–Tucker equations relies on assumptions that the code does not check. For instance, the solution set of equational constraints should be a manifold of dimension $v - e$, where v is the number of variables and e is the number of equations. If this is not satisfied, **KuhnTucker** may not find the extremum.

```
In[11]:= KuhnTucker[x + y, {z == x^2 - x y, z == y^2 + x y, z == x^2 + y^2}, {x, y, z}]
```

```
Out[11]= False
```

Maximize and **Minimize** work in this situation.

```
In[12]:= Minimize[x + y, {z == x^2 - x y, z == y^2 + x y, z == x^2 + y^2}, {x, y, z}]
```

```
Out[12]= {0, {x -> 0, y -> 0, z -> 0}}
```

```
In[13]:= Maximize[x + y, {z == x^2 - x y, z == y^2 + x y, z == x^2 + y^2}, {x, y, z}]
```

```
Out[13]= {0, {x -> 0, y -> 0, z -> 0}}
```

■ Legendre–Gauss Quadrature

Gaussian quadrature gives the best estimate of an integral by picking optimal abscissas, x_i , at which to evaluate the function $f(x)$. It is optimal because the n -point formula is exact for polynomials of degree $\leq 2n - 1$. Legendre–Gauss quadrature (see mathworld.wolfram.com/Legendre-GaussQuadrature.html) is a Gaussian quadrature over the interval $[-1, 1]$ with weighting function $W(x) = 1$, that is,

$$\int_{-1}^1 f(x) dx \approx \sum_{i=1}^n w_i f(x_i),$$

where the abscissas, x_i , for quadrature order n , satisfy $P_n(x_i) = 0$, and the weights are

$$w_i = \frac{2(1 - x_i^2)}{(n+1)^2 P_{n+1}(x_i)^2}.$$

Define the abscissas as the roots of $P_n(x)$ using **Root** and save them as they are computed. This is an *exact* expression for the roots.

```
In[1]:= x /: x_{i..n} := x /: x_{i,n} = Root[Function[x, P_n(x)], i]
```

Define the weights. **RootReduce** expresses the weights as a *single Root* object. Save the weights as they are computed.

$$\text{In}[2]:= w /: w_{i,n} := w /: w_{i,n} = \frac{2(1-x_{i,n}^2)}{(n+1)^2 P_{n+1}(x_{i,n})^2} // \text{RootReduce}$$

Here are the x_i and w_i for $n = 3$.

$$\text{In}[3]:= \text{With}[\{n = 3\}, \text{Table}[\{x_{i,n}, w_{i,n}\}, \{i, n\}]]$$

$$\text{Out}[3]= \begin{pmatrix} -\sqrt{\frac{3}{5}} & \frac{5}{9} \\ 0 & \frac{8}{9} \\ \sqrt{\frac{3}{5}} & \frac{5}{9} \end{pmatrix}$$

Here is x_2 and w_2 for $n = 5$, expressed as radicals.

$$\text{In}[4]:= \text{With}[\{n = 5\}, \{x_{2,n}, w_{2,n}\} // \text{ToRadicals}]$$

$$\text{Out}[4]= \left\{ -\frac{1}{3} \sqrt{\frac{1}{7} (35 - 2\sqrt{70})}, \frac{1}{900} (322 + 13\sqrt{70}) \right\}$$

Abscissas and weights for higher n are better left as **Root** objects (see “Root versus Radicals,” *TMJ*, 9(3), 2005, pp. 535–537). Note that, for fixed n , each w_i can be expressed as a particular root of a polynomial of degree $\lfloor \frac{n}{2} \rfloor$.

Here are the weights for $n = 6$.

$$\text{In}[5]:= \text{With}[\{n = 6\}, \text{Table}[w_{i,n}, \{i, n\}]]$$

$$\begin{aligned} \text{Out}[5]= & \{\text{Root}[2025000\#1^3 - 2025000\#1^2 + 629325\#1 - 58564 \&, 1], \\ & \text{Root}[2025000\#1^3 - 2025000\#1^2 + 629325\#1 - 58564 \&, 2], \\ & \text{Root}[2025000\#1^3 - 2025000\#1^2 + 629325\#1 - 58564 \&, 3], \\ & \text{Root}[2025000\#1^3 - 2025000\#1^2 + 629325\#1 - 58564 \&, 3], \\ & \text{Root}[2025000\#1^3 - 2025000\#1^2 + 629325\#1 - 58564 \&, 2], \\ & \text{Root}[2025000\#1^3 - 2025000\#1^2 + 629325\#1 - 58564 \&, 1] \} \end{aligned}$$

The weights satisfy $\sum_{i=1}^n w_i = 2$. Here we check this for $n = 6$.

$$\text{In}[6]:= \text{Tr}[\%] // \text{RootReduce}$$

$$\text{Out}[6]= 2$$

Now consider computing the integral of a polynomial of degree m , denoted $p_m(x)$.

$$\text{In}[7]:= p_m(x) := a_0 + \sum_{j=1}^m a_j x^j;$$

Compute the integral of $p_7(x)$ over $-1 \leq x \leq 1$.

$$\text{In}[8]:= \int_{-1}^1 p_7(x) dx$$

$$\text{Out}[8]= 2a_0 + \frac{2a_2}{3} + \frac{2a_4}{5} + \frac{2a_6}{7}$$

As expected, an identical answer is obtained using Legendre–Gauss quadrature with $n = 4$ since $2n - 1 = 7$.

```
In[9]:= With[{n = 4}, Sum[w_{i,n} p_7(x_{i,n})] // FullSimplify
```

```
Out[9]= 2 a_0 + \frac{2 a_2}{3} + \frac{2 a_4}{5} + \frac{2 a_6}{7}
```

■ Pascal Matrices

Writing Pascal's triangle as a lower triangular matrix generates one type of *Pascal matrix* (see mathworld.wolfram.com/PascalMatrix.html). Such matrices have many interesting properties (web.mit.edu/18.06/www/pascal-work.pdf).

Define the lower triangular matrix L_n by $\{L_n\}_{i,j} = \binom{i}{j}$, where $i, j = 0, 1, \dots, n - 1$.

```
In[1]:= L_n := Table[Binomial[i, j], {i, 0, n - 1}, {j, 0, n - 1}]
```

Here is L_5 .

```
In[2]:= L_5
```

$$\text{Out[2]} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 2 & 1 & 0 & 0 \\ 1 & 3 & 3 & 1 & 0 \\ 1 & 4 & 6 & 4 & 1 \end{pmatrix}$$

And here is its inverse.

```
In[3]:= L_5^{-1}
```

$$\text{Out[3]} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 \\ -1 & 3 & -3 & 1 & 0 \\ 1 & -4 & 6 & -4 & 1 \end{pmatrix}$$

In general, the inverse of L_n has $\{L_n^{-1}\}_{i,j} = (-1)^{i+j} \binom{i}{j}$.

Clearly $\{L^p\}_{i,j} = p^{i-j} \binom{i}{j}$.

In[4]:= **MatrixPower**[L_5 , p]

$$\text{Out[4]} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ p & 1 & 0 & 0 & 0 \\ p^2 & 2p & 1 & 0 & 0 \\ p^3 & 3p^2 & 3p & 1 & 0 \\ p^4 & 4p^3 & 6p^2 & 4p & 1 \end{pmatrix}$$

Define $v_n(x) = \{x^i\}_{i=0,1,\dots,n}$.

In[5]:= $v_n(\underline{x}) := \underline{x}^{\text{Range}[n]-1}$

Using the binomial theorem, we can show that $L_n^p v_n(x) = v_n(p+x)$, that is,

$$\begin{pmatrix} 1 & & & & \\ p & 1 & & & \\ p^2 & 2p & 1 & & \\ p^3 & 3p^2 & 3p & 1 & \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \begin{pmatrix} 1 \\ x \\ x^2 \\ x^3 \\ \vdots \end{pmatrix} = \begin{pmatrix} 1 \\ p+x \\ (p+x)^2 \\ (p+x)^3 \\ \vdots \end{pmatrix}.$$

“Prove” the binomial theorem.

$$\text{In[6]} := \text{Simplify}[(p+x)^i = \sum_{j=0}^i \binom{i}{j} x^j p^{i-j}, i \in \mathbb{Z}]$$

Out[6]= True

Verify that $L_n^p v_n(x) = v_n(p+x)$ for $n = 5$.

In[7]:= **MatrixPower**[L_5 , p]. $v_5(x)$ // **Factor**

Out[7]= $\{1, p+x, (p+x)^2, (p+x)^3, (p+x)^4\}$

In[8]:= $\% = v_5(p+x)$

Out[8]= True

Writing $L_n^p = e^{A_n p}$, it follows that $A_n = \log(L_n) = \lim_{p \rightarrow 0} (L_n^p - I_n) / p$. Here is A_5 .

$$\text{In[9]} := A_5 = \lim_{p \rightarrow 0} \frac{\text{MatrixPower}[L_5, p] - \text{IdentityMatrix}[5]}{p}$$

$$\text{Out[9]} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 \end{pmatrix}$$

In general, it can be shown that $\{A_n\}_{i,j} = i \delta_{i,j+1}$.

Check that $L_5 = e^{A_5}$.

In[10]:= **MatrixExp**[A_5]

$$\text{Out[10]} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 2 & 1 & 0 & 0 \\ 1 & 3 & 3 & 1 & 0 \\ 1 & 4 & 6 & 4 & 1 \end{pmatrix}$$

Define the upper triangular matrix $\mathcal{U}_n$ by alternating the sign of the rows in L_n^T .

In[11]:= $\mathcal{U}_n := (-1)^{\text{Range}[n]-1} L_n^T$

Here is $\mathcal{U}_5$.

In[12]:= $\mathcal{U}_5$

$$\text{Out[12]} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & -1 & -2 & -3 & -4 \\ 0 & 0 & 1 & 3 & 6 \\ 0 & 0 & 0 & -1 & -4 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$\mathcal{U}_n$ is its own inverse (it is *involutory*).

In[13]:= $\mathcal{U}_5 \cdot \mathcal{U}_5$

$$\text{Out[13]} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Multiplying L_n by its transpose yields a symmetric, positive-definite matrix, denoted S_n , now with Pascal's triangle entries along each skew diagonal (see mathworld.wolfram.com/SkewDiagonal.html).

In[14]:= $S_n := L_n \cdot L_n^T$

In[15]:= S_4

$$\text{Out[15]} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \\ 1 & 3 & 6 & 10 \\ 1 & 4 & 10 & 20 \end{pmatrix}$$

Since $S_n = L_n \cdot L_n^T$ and $\{L_n^T\}_{k,j} = \{L_n\}_{j,k}$, we obtain that

$$\{S_n\}_{i,j} = \sum_{k=0}^i \{L_n\}_{i,k} \{L_n\}_{j,k} = \binom{i+j}{i} = \binom{i+j}{j}.$$

$$\text{In}[16]:= \sum_{k=0}^i \binom{i}{k} \binom{j}{k}$$

$$\text{Out}[16]= \frac{\Gamma(i+j+1)}{\Gamma(i+1)\Gamma(j+1)}$$

$$\text{In}[17]:= \text{FullSimplify}[\% = \binom{i+j}{i} = \binom{i+j}{j}]$$

$$\text{Out}[17]= \text{True}$$

The inverse of S_n has integer entries.

$$\text{In}[18]:= S_5^{-1}$$

$$\text{Out}[18]= \begin{pmatrix} 5 & -10 & 10 & -5 & 1 \\ -10 & 30 & -35 & 19 & -4 \\ 10 & -35 & 46 & -27 & 6 \\ -5 & 19 & -27 & 17 & -4 \\ 1 & -4 & 6 & -4 & 1 \end{pmatrix}$$

It can be shown that $S_n^{-1} = \mathcal{U}_n^{-1} \cdot S_n \cdot \mathcal{U}_n$. That is S_n^{-1} is similar to S_n (see mathworld.wolfram.com/SimilarMatrices.html).

Verify that $S_n^{-1} = \mathcal{U}_n^{-1} \cdot S_n \cdot \mathcal{U}_n$ for $n = 5$.

$$\text{In}[19]:= S_5^{-1} = \mathcal{U}_5^{-1} \cdot S_5 \cdot \mathcal{U}_5$$

$$\text{Out}[19]= \text{True}$$

Since similar matrices have the same eigenvalues, the eigenvalues of S_n must come in reciprocal pairs, λ and λ^{-1} . Moreover, if n is odd, then the “middle” eigenvalue must be unity.

Here are the eigenvalues of S_3 .

$$\text{In}[20]:= \text{Eigenvalues}[S_3]$$

$$\text{Out}[20]= \{4 + \sqrt{15}, 1, 4 - \sqrt{15}\}$$

Confirm that the eigenvalues come in reciprocal pairs.

$$\text{In}[21]:= 1 / \%$$

$$\text{Out}[21]= \left\{ \frac{1}{4 + \sqrt{15}}, 1, \frac{1}{4 - \sqrt{15}} \right\}$$

$$\text{In}[22]:= \text{RootReduce}[\%]$$

$$\text{Out}[22]= \{4 - \sqrt{15}, 1, 4 + \sqrt{15}\}$$

■ Generalized Padé Approximation

It is often the case that, while an exact solution is difficult or impossible, series or asymptotic solutions to a particular problem can be obtained without too much difficulty. As is well known, Padé approximations (see mathworld.wolfram.com/PadeApproximant.html) are usually superior to Taylor expansions when functions contain poles, because the use of rational functions allows them to be well represented.

Define $P_N(x)$ and $Q_M(x)$ as polynomials of degrees N and M , with $Q_M(0) = 1$.

$$\text{In}[1]:= P_{\text{n_}}(\text{x_}) := \sum_{i=0}^n p_i x^i$$

$$\text{In}[2]:= Q_{\text{m_}}(\text{x_}) := 1 + \sum_{i=1}^m q_i x^i$$

The coefficients in the $[N, M]$ Padé approximant to $f(x)$ are determined by requiring that the Taylor expansion of $f(x) - P_N(x)/Q_M(x)$ is of order $O(x^{N+M+1})$.

Suppose that you have obtained the following Taylor series up to $O(x^5)$ for a function of interest.

$$\text{In}[3]:= \text{taylor}[\text{x_}] = 1 - \frac{2x}{3} + \frac{4x^2}{15} - \frac{8x^3}{105} + \frac{16x^4}{945};$$

Here is the $[2, 2]$ Padé approximant for this series.

$$\text{In}[4]:= \ll \text{Calculus`}$$

$$\text{In}[5]:= \text{pade}[\text{x_}] = \text{Pade}[\text{taylor}[\text{x}], \{\text{x}, 0, 2, 2\}]$$

$$\text{Out}[5]= \frac{\frac{32x^2}{945} - \frac{2x}{9} + 1}{\frac{4x^2}{63} + \frac{4x}{9} + 1}$$

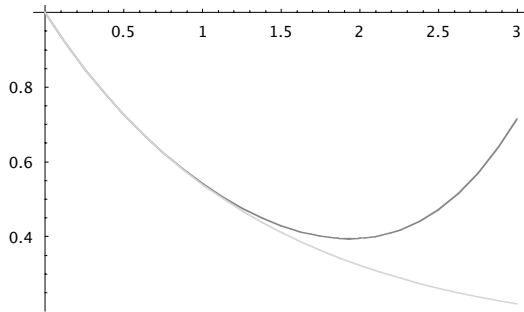
Verify that the Taylor expansion of $f(x) - P_2(x)/Q_2(x)$ is of order $O(x^5)$.

$$\text{In}[6]:= \text{taylor}[\text{x}] - \text{pade}[\text{x}] + O[\text{x}]^5$$

$$\text{Out}[6]= O(x^5)$$

A plot shows that the Taylor series (—) diverges from the Padé approximant (—) near $x = 1$.

```
In[7]:= small = Plot[{taylor[x], pade[x]}, {x, 0, 3}, PlotStyle -> {Hue[1], Hue[ $\frac{1}{3}$ ]}
```



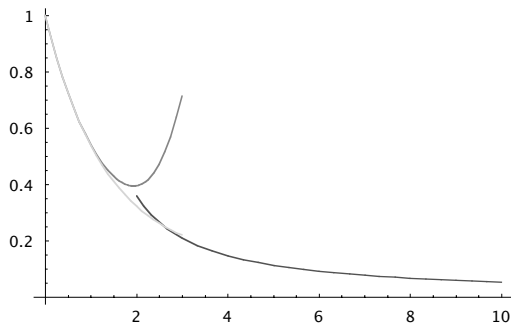
Suppose now that you have also computed the following asymptotic series for the same function.

```
In[8]:= asymptotic[x_] =  $\frac{1}{2x} + \frac{1}{4x^2} + \frac{3}{8x^3}$ ;
```

A plot shows that the Padé approximant about $x = 0$ (—) and the asymptotic expansion (---) intersect for $x \approx 2.5$.

```
In[9]:= << Graphics`
```

```
In[10]:= large = DisplayTogether[
    Plot[asymptotic[x], {x, 2, 10}, PlotStyle -> Hue[ $\frac{2}{3}$ ]], small]
```



How can you incorporate information from expansions of a function about two or more points? One approach is to generalize the idea of computing Padé approximants [1]. Suppose $f(x)$ has the asymptotic expansions

$$f(x) \sim \sum_{i=0}^{\infty} a_i (x - x_0)^i, \quad x \rightarrow x_0,$$

$$f(x) \sim \sum_{i=0}^{\infty} b_i (x - x_1)^i, \quad x \rightarrow x_1,$$

in the neighborhoods of distinct points x_0 and x_1 , respectively. To compute a two-point Padé approximant to $f(x)$, we require that the rational function $P_N(x)/Q_M(x)$ agrees with $f(x)$ up to order J about x_0 and to order K about x_1 , where $J + K = N + M + 1$.

To use all the information at hand, note that for $x_0 = 0$, $J = 5$, and for $x_1 = \infty$, $K = 4$.

`In[11]:= j = Exponent[taylor[x], x] + 1`

`Out[11]= 5`

`In[12]:= k = Exponent[asymptotic[x], 1/x] + 1`

`Out[12]= 4`

The diagonal Padé approximant then has $N = M = 4$.

`In[13]:= n = m = 1/2 (j + k - 1)`

`Out[13]= 4`

It is straightforward to determine the coefficients p_i and q_i in terms of a_i and b_i . Here we obtain these coefficients using series arithmetic via **Solve**.

`In[14]:= Solve[`

$$\left\{ \text{taylor}[x] - \frac{P_n(x)}{Q_m(x)} + O[x]^j = 0, \text{asymptotic}[x] - \frac{P_n(x)}{Q_m(x)} + O[x, \infty]^k = 0 \right\},$$

`Join[Table[pi, {i, 0, n}], Table[qi, {i, m}]]]`

`Out[14]= {{p2 → 52/945, p3 → 8/945, p1 → 2/9, p4 → 0,`

$$p_0 \rightarrow 1, q_2 \rightarrow \frac{8}{21}, q_3 \rightarrow \frac{32}{315}, q_1 \rightarrow \frac{8}{9}, q_4 \rightarrow \frac{16}{945}}\}}$$

Here is the two-point Padé approximant to $f(x)$.

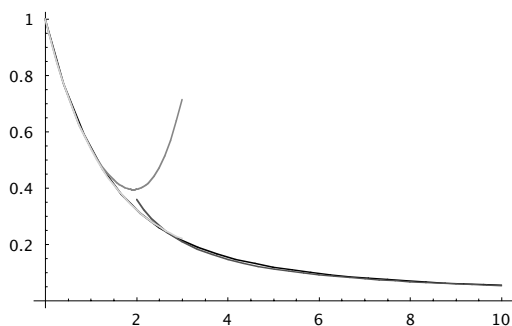
`In[15]:= twopoint[x_] = Pn(x)/Qm(x) /. First[%]`

`Out[15]=`

$$\frac{\frac{8x^3}{945} + \frac{52x^2}{945} + \frac{2x}{9} + 1}{\frac{16x^4}{945} + \frac{32x^3}{315} + \frac{8x^2}{21} + \frac{8x}{9} + 1}$$

The two-point Padé approximant (—) agrees with the Taylor series about $x = 0$ (—), the one-point Padé approximant about $x = 0$ (—), and the asymptotic expansion (—) about $x = \infty$.

`In[16]:= DisplayTogether[Plot[twopoint[x], {x, 0, 10}], large]`



■ References

- [1] C. M. Bender and S. A. Orszag, *Advanced Mathematical Methods for Scientists and Engineers*, New York: McGraw-Hill, 1978, pp. 393–395.

Paul Abbott

School of Physics, M013
University of Western Australia
35 Stirling Highway
Crawley WA 6009, Australia
tmj@physics.uwa.edu.au
physics.uwa.edu.au/~paul

In and Out

Edited by Paul Abbott

In and Out offers readers an opportunity to ask questions of the experts. *The Mathematica Journal* encourages readers to submit problems in care of the editor. Answers posted to the *Mathematica* newsgroup, comp.soft-sys.math.mathematica, that appear here are edited for clarity and length.

■ Closed Form Integral

In the last column, a closed form for the integral

$$a_{k,l} = \int_0^1 P_k(x) P_l(2x-1) dx$$

was obtained. Here was the result, with the denominator simplified slightly by noting that $(2l+1)(1)_l(l+1)_l = (2l+1)!$.

$$a_{k,l} = \frac{1}{(2l+1)!} ((-1)^l 2^{-k} {}_2F_1(l-k, -k+l+1; 2l+2; -1) (-k)_l (k+1)_l);$$

Elmar Zeitler (zeitler@fhi-berlin.mpg.de) suggested an alternative approach: expanding $P_k(x)$ as a Gaussian hypergeometric function (functions.wolfram.com/Polynomials/LegendreP/26/01/01/0001),

$$P_k(x) = {}_2F_1\left(-k, k+1; 1; \frac{1-x}{2}\right),$$

and using functions.wolfram.com/05.03.07.0004.01 to write

$$P_l(2x-1) = \frac{1}{l!} \frac{\partial^l (x^2-x)^l}{\partial x^l},$$

one has to compute

$$\int_0^1 (1-x)^m \frac{1}{l!} \frac{\partial^l (x^2-x)^l}{\partial x^l} dx,$$

which can be done using integration by parts. Alternatively, evaluating the integral for $l = 0, 1, \dots, 4$, the pattern is clear.

In[2]:= Simplify[Table[{l, $\int_0^1 (1-x)^m \frac{1}{l!} \frac{\partial^l (x^2-x)^l}{\partial x^l} dx$ }, {l, 0, 4}], m ≥ 0] // Factor

$$\text{Out[2]} = \begin{pmatrix} 0 & \frac{1}{m+1} \\ 1 & -\frac{m}{(m+1)(m+2)} \\ 2 & \frac{(m-1)m}{(m+1)(m+2)(m+3)} \\ 3 & -\frac{(m-2)(m-1)m}{(m+1)(m+2)(m+3)(m+4)} \\ 4 & \frac{(m-3)(m-2)(m-1)m}{(m+1)(m+2)(m+3)(m+4)(m+5)} \end{pmatrix}$$

One observes that

$$\int_0^1 (1-x)^m \frac{1}{l!} \frac{\partial^l (x^2-x)^l}{\partial x^l} dx = \frac{(-m)_l}{(m+1)_{l+1}},$$

leading to an alternative form for $a_{k,l}$, denoted $b_{k,l}$.

In[3]:= $b_{k,l} = \sum_{m=0}^k \frac{(-k)_m (k+1)_m}{(1)_m m! 2^m} \frac{(-m)_l}{(m+1)_{l+1}}$ // FullSimplify

$$\text{Out[3]} = (-1)^l {}_3\tilde{F}_2\left(1, -k, k+1; 1-l, l+2; \frac{1}{2}\right)$$

Since $a_{k,l} = 0$ for $l > k$, we check that $a_{k,l} = b_{k,l}$ for $0 \leq l \leq k \leq 4$.

In[4]:= Table[{k, l, $a_{k,l} = b_{k,l}$ }, {k, 0, 4}, {l, 0, k}]

$$\text{Out[4]} = \left\{ (0 \ 0 \ \text{True}), \begin{pmatrix} 1 & 0 & \text{True} \\ 1 & 1 & \text{True} \end{pmatrix}, \begin{pmatrix} 2 & 0 & \text{True} \\ 2 & 1 & \text{True} \\ 2 & 2 & \text{True} \end{pmatrix}, \begin{pmatrix} 3 & 0 & \text{True} \\ 3 & 1 & \text{True} \\ 3 & 2 & \text{True} \\ 3 & 3 & \text{True} \end{pmatrix}, \begin{pmatrix} 4 & 0 & \text{True} \\ 4 & 1 & \text{True} \\ 4 & 2 & \text{True} \\ 4 & 3 & \text{True} \\ 4 & 4 & \text{True} \end{pmatrix} \right\}$$

Alternatively, expanding both $P_k(x)$ and $P_l(2x-1) = (-1)^l P_l(1-2x)$ as Gaussian hypergeometric functions, the result is immediate.

$$\text{In[5]} := \sum_{m=0}^k \frac{(-k)_m (k+1)_m}{(1)_m m! 2^m} \sum_{n=0}^l \frac{(-l)_n (l+1)_n}{(1)_n n!}$$

Assuming[$m \geq 0 \wedge n \geq 0$, $(-1)^l \int_0^1 (1-x)^m x^n dx$] // FullSimplify

$$\text{Out[5]} = (-1)^l {}_3\tilde{F}_2\left(1, -k, k+1; 1-l, l+2; \frac{1}{2}\right)$$

In other words, we have demonstrated the identity

$$\frac{{}_2^{-k} {}_2F_1(l-k, -k+l+1; 2l+2; -1) (-k)_l (k+1)_l}{(2l+1)!} = {}_3\tilde{F}_2\left(1, -k, k+1; 1-l, l+2; \frac{1}{2}\right).$$

■ Sum of Gaussians

Q: I would like to find a closed form for the sum

$$\text{In}[1]:= f(z) = \sum_{n=-\infty}^{\infty} e^{-\frac{1}{2}(z+n)^2}$$

$$\text{Out}[1]= \sum_{n=-\infty}^{\infty} e^{-\frac{1}{2}(n+z)^2}$$

Clearly $f(z)$ is periodic with unit period (put $z \rightarrow z + 1$ and $n \rightarrow n - 1$). How can I compute the Fourier series expansion of $f(z)$?

A: Putting $\tau \rightarrow 2i\pi$ and $z \rightarrow \pi z$ into the formula functions.wolfram.com/09.03.06.0019.01,

$$\vartheta_3(z, q) = \frac{\sqrt{i}}{\sqrt{\tau}} \sum_{n=-\infty}^{\infty} e^{-\frac{\pi i}{\tau} (n + \frac{z}{\pi})^2} ; q = e^{i\pi\tau},$$

that is,

$$\text{In}[2]:= \left\{ \frac{\sqrt{i}}{\sqrt{\tau}} e^{-\frac{\pi i}{\tau} (n + \frac{z}{\pi})^2}, q = e^{i\pi\tau} \right\} /. \{ \tau \rightarrow 2i\pi, z \rightarrow \pi z \} // \text{FullSimplify}$$

$$\text{Out}[2]= \left\{ \frac{e^{-\frac{1}{2}(n+z)^2}}{\sqrt{2\pi}}, q = e^{-2\pi^2} \right\}$$

we can express the sum in terms of **EllipticTheta** as follows:

$$\sum_{n=-\infty}^{\infty} e^{-\frac{1}{2}(n+z)^2} = \sqrt{2\pi} \vartheta_3(\pi z, e^{-2\pi^2}).$$

The Fourier series expansion of $\vartheta_3(z, q)$ (essentially its standard definition) is given at functions.wolfram.com/09.03.06.0001.01,

$$\vartheta_3(z, q) = 1 + 2 \sum_{k=1}^{\infty} q^{k^2} \cos(2kz) ; |q| < 1.$$

Here are the first few terms of the Fourier series expansion of $f(z)$.

$$\text{In}[3]:= \sqrt{2\pi} \left(1 + 2 \sum_{k=1}^3 q^{k^2} \cos(2kz) \right) /. \{ z \rightarrow \pi z, q \rightarrow e^{-2\pi^2} \}$$

$$\text{Out}[3]= \sqrt{2\pi} (2 (e^{-2\pi^2} \cos(2\pi z) + e^{-8\pi^2} \cos(4\pi z) + e^{-18\pi^2} \cos(6\pi z)) + 1)$$

Note that the Fourier coefficients decrease extremely rapidly.

```

In[4]:= qRange[3]^2 /. q -> e-2 π^2
Out[4]= {e-2 π^2, e-8 π^2, e-18 π^2}
In[5]:= N[%]
Out[5]= {2.67529 × 10-9, 5.1225 × 10-35, 7.01996 × 10-78}

```

An excellent approximation is obtained by keeping just the first term in the sum, that is, $f(z) \approx \sqrt{2\pi} (1 + 2 e^{-2\pi^2} \cos(2\pi z))$.

■ *k*-ary Divisors

Q: The *k*-ary divisors of *n*, written $d \mid_k n$, are defined at mathworld.wolfram.com/k-aryDivisor.html. How can I implement $d \mid_k n$?

A: A divisor of a number *n* is a number *d* which divides *n* (written $d \mid n$), also called a factor. The divisors *d* of *n*, that is $d \mid n$, are computed using **Divisors**. Here are the divisors of 12.

```

In[1]:= Divisors[12]
Out[1]= {1, 2, 3, 4, 6, 12}

```

It is straightforward to implement the notation $d \mid n$.

```

In[2]:= << Utilities`Notation`
In[3]:= Notation[d_ | n_ => MemberQ[Divisors[n], d_]]

```

Clearly 3 divides 12.

```

In[4]:= 3 | 12
Out[4]= True

```

A 1-ary or *unitary* divisor (mathworld.wolfram.com/UnitaryDivisor.html) *d* of *n* satisfies $d \perp n/d$, that is *d* is relatively prime to *n/d* or $\gcd(d, n/d) = 1$. **divisors[1] [n]** computes, and caches, the list of unitary divisors of *n*, $d \mid_1 n$.

```

In[5]:= divisors[1] [n_Integer] :=
      divisors[1] [n] = Select[Divisors[n], gcd(#[1], n/#[1]) == 1 &]

```

Here are the unitary divisors of 12.

```

In[6]:= divisors[1] [12]
Out[6]= {1, 3, 4, 12}

```

Define the notation $d \mid_k n$.

`In[7]:= Notation[d_ |k n_  $\Rightarrow$  MemberQ[divisors[k_][n_], d_]]`

In this notation, a divisor d of n is written $d \mid_0 n = d \mid n$.

`In[8]:= divisors[0][n_Integer] := Divisors[n]`

Here are the unitary divisors for the first few integers (Sloane's A077610).

`In[9]:= Table[{n, divisors[1][n]}, {n, 12}]`

`Out[9]=`

1	{1}
2	{1, 2}
3	{1, 3}
4	{1, 4}
5	{1, 5}
6	{1, 2, 3, 6}
7	{1, 7}
8	{1, 8}
9	{1, 9}
10	{1, 2, 5, 10}
11	{1, 11}
12	{1, 3, 4, 12}

For a prime power p^y , the unitary divisors are 1 and p^y [1].

`In[10]:= {#, divisors[1][#]} & /@ Prime[Range[10]]`

`Out[10]=`

2	{1, 2}
3	{1, 3}
5	{1, 5}
7	{1, 7}
11	{1, 11}
13	{1, 13}
17	{1, 17}
19	{1, 19}
23	{1, 23}
29	{1, 29}

2 is a divisor but not a unitary divisor of 12.

`In[11]:= {2 |0 12, 2 |1 12}`

`Out[11]= {True, False}`

d is called a k -ary divisor of n , written $d \mid_k n$, if the greatest common $(k-1)$ -ary divisor of d and n/d is 1. Recursive implementation, using caching, is straightforward.

`In[12]:= divisors[k_Integer][n_Integer] := divisors[k][n] =`

`Select[Divisors[n], Max[divisors[k-1][#]  $\cap$  divisors[k-1][ $\frac{n}{\#}$ ]] == 1 &]`

Here are the 2-ary or *biunitary divisors* for the first few integers.

`In[13]:= Table[{n, divisors[2][n]}, {n, 12}]`

`Out[13]=`

$$\begin{pmatrix} 1 & \{1\} \\ 2 & \{1, 2\} \\ 3 & \{1, 3\} \\ 4 & \{1, 4\} \\ 5 & \{1, 5\} \\ 6 & \{1, 2, 3, 6\} \\ 7 & \{1, 7\} \\ 8 & \{1, 2, 4, 8\} \\ 9 & \{1, 9\} \\ 10 & \{1, 2, 5, 10\} \\ 11 & \{1, 11\} \\ 12 & \{1, 3, 4, 12\} \end{pmatrix}$$

For a prime power p^y , the biunitary divisors are $1, p, p^2, \dots, p^y$ except for $p^{y/2}$ when y is even [1].

`In[14]:= Table[{n, 3^n |_2 3^6}, {n, 0, 6}]`

`Out[14]=`

$$\begin{pmatrix} 0 & \text{True} \\ 1 & \text{True} \\ 2 & \text{True} \\ 3 & \text{False} \\ 4 & \text{True} \\ 5 & \text{True} \\ 6 & \text{True} \end{pmatrix}$$

Here are the k -ary divisors of 3^6 for $k = 0, 1, \dots, 10$. One observes that a fixed point is attained.

`In[15]:= Table[{k, divisors[k][3^6]}, {k, 0, 8}]`

`Out[15]=`

$$\begin{pmatrix} 0 & \{1, 3, 9, 27, 81, 243, 729\} \\ 1 & \{1, 729\} \\ 2 & \{1, 3, 9, 81, 243, 729\} \\ 3 & \{1, 9, 81, 729\} \\ 4 & \{1, 3, 9, 81, 243, 729\} \\ 5 & \{1, 9, 81, 729\} \\ 6 & \{1, 9, 81, 729\} \\ 7 & \{1, 9, 81, 729\} \\ 8 & \{1, 9, 81, 729\} \end{pmatrix}$$

A divisor d of n is called *infinitary*, $d \mid_{\infty} n$, if it is a product of divisors of the form $p^{y_i 2^i}$, where p^y is a prime power dividing n and $\sum_i y_i 2^i$ is the binary representation of y (see Sloane's A037445). The infinitary divisors are then those factors d

that have zeros in the binary representation of all y_i s where n itself does. This can be directly coded using **BitOr**.

```

In[16]:= y[n_] := Select[Range[0, n], BitOr[n, #] == n &]

In[17]:= divisors[∞][n_] := Sort@Flatten@Outer[Times,
      Sequence @@ (FactorInteger[n] /. {p_, m_Integer} :> py[m])]

In[18]:= divisors[∞][1] := {1}

 $p^x$  is an infinitary divisor of  $p^y$  (with  $y > 0$ ) if  $p^x \mid_{y-1} p^y$  [1].

In[19]:= 113 |6 117
Out[19]= True

In[20]:= 113 |∞ 117
Out[20]= True

```

Here are the infinitary divisors for the first few integers.

```

In[21]:= Table[{n, divisors[∞][n]}, {n, 16}]

```

```

Out[21]= {
  {1, {1}},
  {2, {1, 2}},
  {3, {1, 3}},
  {4, {1, 4}},
  {5, {1, 5}},
  {6, {1, 2, 3, 6}},
  {7, {1, 7}},
  {8, {1, 2, 4, 8}},
  {9, {1, 9}},
  {10, {1, 2, 5, 10}},
  {11, {1, 11}},
  {12, {1, 3, 4, 12}},
  {13, {1, 13}},
  {14, {1, 2, 7, 14}},
  {15, {1, 3, 5, 15}},
  {16, {1, 16}}
}

```

Taking the logarithm base p of the infinitary divisors of p^y gives the x values for which p^x are infinitary divisors of p^y .

```

In[22]:= log2(divisors[∞] /@ 2Range[7])

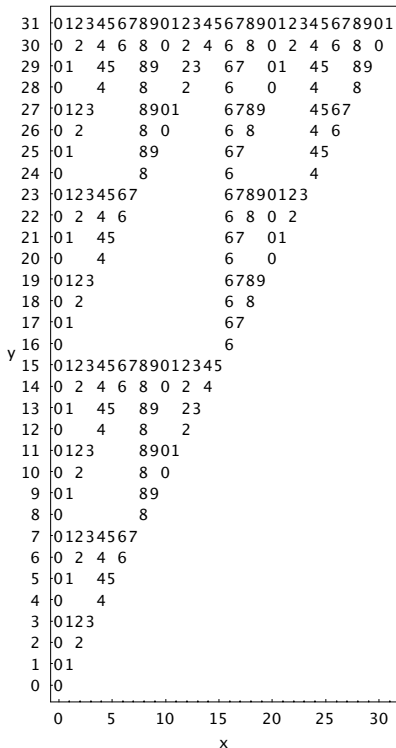
Out[22]= {{0, 1}, {0, 2}, {0, 1, 2, 3}, {0, 4}, {0, 1, 4, 5}, {0, 2, 4, 6}, {0, 1, 2, 3, 4, 5, 6, 7}}

```

Here is a plot of the infinitary divisors of p^y , computed with $p = 2$ but valid for arbitrary $p \in \mathbb{P}$, for $y = 0, 1, \dots, 31$. Showing the last (decimal) digit of x enables the actual divisors to be read off easily [1].

```
In[23]:= Show[
  Graphics[MapIndexed[Text[Last[IntegerDigits[#1]], {#1, First[#2] - 1}] &,
    log2(divisors[∞])/@ 2Range[0,31]), {2}]], Frame → True,
  FrameTicks → {Automatic, Range[0, 31], None, None}, AspectRatio → 2,
  TextStyle → {FontFamily → "MR", FontSize → 6}, FrameLabel → {x, y}]
```

From In[23]:=



The appearance of the Sierpinski sieve fractal mathworld.wolfram.com/SierpinskiSieve.html is interesting.

■ Asymptotic Expansion of a Sum

Q: Let r be an arbitrary irrational number. How can I find an asymptotic approximation to

$$f(x) = \sum_{n=1}^{\infty} \frac{x^n}{n-r} \quad (1)$$

as x approaches 1 from below?

A: The sum can be expressed as a Gaussian hypergeometric function.

$$\text{In}[1]:= f[r_][x_]=\sum_{n=1}^{\infty}\frac{x^n}{n-r}$$

$$\text{Out}[1]=-\frac{x {}_2F_1(1-r, 1; 2-r; x)}{r-1}$$

There are at least three advantages of this representation:

1. *Mathematica* includes arbitrary-precision algorithms for computing this function. For example, here is the sum with $r = \sqrt{541}$ and $x = 1 - 10^{-3}$.

$$\text{In}[2]:= N[f[\sqrt{541}][1 - 10^{-3}], 20]$$

$$\text{Out}[2]= 0.21580785006326598375$$

2. Asymptotic approximations can be computed via series expansion. Introducing $x = 1 - \epsilon$, the following expansion is useful when $r\epsilon \ll 1$.

$$\text{In}[3]:= f[r][1 - \epsilon] + O[\epsilon]^3 // \text{FullSimplify}$$

$$\text{Out}[3]= (-H_{-r} - \log(\epsilon)) + r(H_{-r} + \log(\epsilon) - 1)\epsilon - \frac{1}{4}((r-1)r(2H_{1-r} + 2\log(\epsilon) - 3))\epsilon^2 + O(\epsilon^3)$$

Here is the asymptotic approximation with $r = \sqrt{541}$ and $\epsilon = 10^{-3}$.

$$\text{In}[4]:= g[r_][\epsilon_] = \text{Normal}[\%];$$

$$\text{In}[5]:= g[\sqrt{541.}][10^{-3}]$$

$$\text{Out}[5]= 0.215812$$

3. The asymptotic series expansion in closed form can be obtained using functions.wolfram.com/07.23.06.0012.01,

$${}_2F_1(a, b; a+b; z) =$$

$$\frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \sum_{k=0}^{\infty} \frac{(a)_k (b)_k}{k!^2} (2\psi(k+1) - \psi(a+k) - \psi(b+k) - \log(1-z))(1-z)^k /;$$

$$|1-z| < 1.$$

Here is the result.

$$\text{In}[6]:= f[r][1 - \epsilon] /.$$

$${}_2F_1(a_ , b_ ; c_ ; z_) \rightarrow \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \sum_{k=0}^{\infty} \frac{(a)_k (b)_k}{k!^2} (-\log(1-z) + 2\psi(k+1) - \psi(a+k) - \psi(b+k))(1-z)^k /; c == a+b // \text{FullSimplify}$$

$$\text{Out}[6]= -(\epsilon - 1) \sum_{k=0}^{\infty} \frac{\epsilon^k (1)_k (1-r)_k (-\log(\epsilon) + \psi^{(0)}(k+1) - \psi^{(0)}(k-r+1))}{(k!)^2}$$

This can be used to compute the number of asymptotic terms required. For example, $k_{\max} = 5$ terms are required to get a result accurate to at least 10 decimal places with $r = \sqrt{541}$ and $\epsilon = 10^{-3}$.

```

In[7]:= With[{ $\epsilon = 10^{-3}$ ,  $r = \sqrt{541}$ .}],
Table[{ $k$ ,  $\frac{1}{(k!)^2} (\epsilon^k (1)_k (1-r)_k (-\log(\epsilon) + \psi^{(0)}(k+1) - \psi^{(0)}(k-r+1)))$ },
{ $k$ , 0, 5}]]

Out[7]=  $\begin{pmatrix} 0 & 0.24429 \\ 1 & -0.0286972 \\ 2 & 0.000434477 \\ 3 & -3.54558 \times 10^{-6} \\ 4 & 1.93943 \times 10^{-8} \\ 5 & -7.79834 \times 10^{-11} \end{pmatrix}$ 

```

■ Variable Order Simplify

Q: Suppose I have variables that are related in the following way: $d = a + b$ and $e = c + d$. Is there a straightforward way to get **FullSimplify** to consider these relationships, returning the simplest possible expression in terms of a , b , c , d , and e , when performing simplification?

A: Adam Strzebonski (adams@wolfram.com) answers: **VOISimplify**, a function for variable-order independent simplification that uses **FullSimplify** and considers all permutations of the variables, does this.

```

In[1]:= VOISimplify(vars_, expr_, assum_ : True) :=
Module[{perm, ee, best}, perm = Permutations[vars];
ee = (FullSimplify @@ ({expr, assum} /. Thread[vars -> #1]) &) /@ perm;
best = Sort[{LeafCount /@ ee, ee, perm}][[1]];
best[[2]] /. Thread[best[[3]] -> vars]]

```

Clearly, $d - b = a$.

```

In[2]:= VOISimplify({a, b, c, d, e}, d - b, {d = a + b, e = c + d})

```

```

Out[2]= a

```

Here is a more complicated example.

```
In[3]:= VOISimplify({a, b, c, d, e},
      a2 + 2 b a + 2 c a + b2 + c2 + d2 + e2 + 2 b c + 2 c e - 2 d e, {d = a + b, e = c + d})
Out[3]= (c + e)2
```

■ Simultaneous Recurrence Relations

Q : I have a set of recurrence relations, say

$$\begin{aligned} a_n &= 4 a_{n-1} (1 - a_{n-1}), \\ b_n &= 4 b_{n-1} (1 - b_{n-1}), \\ c_n &= \frac{1}{2} (a_n + b_n). \end{aligned}$$

How can I simplify them into a single recurrence relation involving only $c_n, c_{n-1}, c_{n-2}, \dots$?

A : Daniel Lichtblau (danl@wolfram.com) answers: To eliminate a_n, a_{n-1}, b_n , and b_{n-1} , you need more equations. Write these equations as polynomials, including the polynomial for c_{n-1} .

```
In[1]:= polys[n_] = {an - 4 an-1 (1 - an-1),
      bn - 4 bn-1 (1 - bn-1), cn -  $\frac{1}{2}$  (an + bn), cn-1 -  $\frac{1}{2}$  (an-1 + bn-1)};
```

To acquire sufficiently many polynomials to eliminate all a_i and b_i variables, observe that putting $n \rightarrow n - 1$ gives a total of 7 polynomials in 9 variables.

```
In[2]:= allpolys = Union[polys[n], polys[n - 1]]
Out[2]= {an-1 - 4 (1 - an-2) an-2, an - 4 (1 - an-1) an-1,
      bn-1 - 4 (1 - bn-2) bn-2, bn - 4 (1 - bn-1) bn-1,
       $\frac{1}{2}$  (-an-2 - bn-2) + cn-2,  $\frac{1}{2}$  (-an-1 - bn-1) + cn-1,  $\frac{1}{2}$  (-an - bn) + cn}
In[3]:= Length[allpolys]
Out[3]= 7
In[4]:= vars = Variables[allpolys]
Out[4]= {an-2, an-1, an, bn-2, bn-1, bn, cn-2, cn-1, cn}
In[5]:= Length[vars]
Out[5]= 9
```

We want to eliminate 6 variables, **elims**, from the 7 polynomials.

```

In[6]:= cs = Cases[vars, c_]
Out[6]= {cn-2, cn-1, cn}
In[7]:= elims = Complement[vars, cs]
Out[7]= {an-2, an-1, an, bn-2, bn-1, bn}
Here is the desired relation.
In[8]:= Solve[Thread[allpolys == 0], cn, elims] // FullSimplify
Out[8]= {{cn → 64 (cn-2 - 1) cn-2 (1 - 2 cn-2)2 - 4 cn-12 + 4 (16 (cn-2 - 1) cn-2 + 5) cn-1}}
```

■ Aborting Messages

Q: I would like to abort a computation whenever an error occurs, avoiding evaluation of the subsequent expressions within a cell. For example, in the following computation, I would like to avoid computing $N[\zeta(2000)]$ after $\sin(2, 2)$ generates an error message.

```
In[1]:= sin(2, 2); N[ζ(2000)]
```

Sin::argx : Sin called with 2 arguments; 1 argument is expected. More...

```
Out[1]= 1.
```

One can immediately abort a computation by replacing the default for **\$Messages** with a call to **Abort**.

```

In[2]:= default = $Messages
Out[2]= {OutputStream[stdout, 1]}
In[3]:= $Messages := Abort[]
```

However, now when an error occurs, the computation is aborted without any message.

```

In[4]:= sin(2, 2); N[ζ(2000)]
Out[4]= $Aborted
```

Is there a way to display the message and then abort?

A: Omega Consulting (omegaconsultinggroup.com) answers: Using **\$Messages := Abort[]** is quite a creative trick. Unfortunately, you cannot use it and also get the message. **\$Messages** is a list of **OutputStreams** that any messages are written to. First, let us restore **\$Messages** to its default value.

```
In[5]:= $Messages = default;
```

If the **OutputStream** is **stdout**, then you get the normal message. If it is an **OutputStream** to a file, then it is written to that file. You might think that the following would work:

```
In[6]:= $Messages := {OutputStream["stdout", 1], Abort[]}
```

However, it does not.

```
In[7]:= sin(2, 2); N[ζ(2000)]
```

```
Out[7]= $Aborted
```

Why not? Because **\$Messages** is evaluated *before* the messages are actually sent to the streams. Again, we restore **\$Messages** to its default value.

```
In[8]:= $Messages = default;
```

Still there is another trick we can use, called a trap. A message is created whenever the **Message** function is called. We can redefine this function by unprotecting it.

```
In[9]:= Unprotect[Message];
```

Now we create a global variable, **\$AbortMessage**, to indicate what we want to do with a **Message**. When the variable is **True**, we want to deviate from the default behavior. When **False**, we want the default behavior (i.e., a message).

```
In[10]:= $AbortMessage = True;
```

Now we add a definition for when the variable is **True**. It calls **Message** with the variable set to **False** (creating a message) and then aborts.

```
In[11]:= Message[args___] :=  
  (Block[{$AbortMessage = False}, Message[args]]; Abort[]) /; $AbortMessage
```

To be safe we reprotect **Message**.

```
In[12]:= Protect[Message];
```

Now we try it.

```
In[13]:= sin(2, 2); N[ζ(2000)]
```

Sin::argx : Sin called with 2 arguments; 1 argument is expected. More...

```
Out[13]= $Aborted
```

■ Notebook Options

Q : How do you add a new option to a **Notebook[]**?

A: John Fultz (jfulzt@wolfram.com) answers: Use **SetOptions**. It is a good idea to store information using the **TaggingRules** option, which is specifically designated for this kind of use. For example,

```
In[1]:= SetOptions[InputNotebook[],
          TaggingRules → {NewOption → "new option string"}]

In[2]:= NewOption /. (TaggingRules /. Options[InputNotebook[], TaggingRules])

Out[2]= new option string
```

If you tried the **SetOptions** syntax directly on the option, it would work, but in a slightly unexpected way.

```
In[3]:= SetOptions[InputNotebook[], NewOption → "new option string"]
```

You will hear a beep. No result is returned for the usual syntax.

```
In[4]:= Options[InputNotebook[], NewOption]
```

Options::optnf : NewOption is not a known option for NotebookObject. More...

```
Out[4]= {}
```

However, the option is being saved.

```
In[5]:= NewOption /. Options[InputNotebook[]]

Out[5]= new option string
```

So, what is going on here? This is using a mechanism for forward compatibility. For example, what if a notebook from a future version with unknown options were to be read by an earlier version? It should just preserve options it does not understand. So, that is exactly what it is doing. **Options** on that specific option will not work because, internally, the option has been saved as a hidden option, but that hidden option always writes its contents out to files and the general **Options** requests.

■ References

- [1] G. L. Cohen, "On an Integer's Infinitary Divisors," *Mathematics of Computation*, **54**(189), 1990, pp. 395–411.

Paul Abbott

School of Physics, M013
The University of Western Australia
35 Stirling Highway
Crawley WA 6009, Australia
tmj@physics.uwa.edu.au
physics.uwa.edu.au/~paul

Mathematical Searching of The Wolfram Functions Site

Michael Trott

The Wolfram Functions Site functions.wolfram.com contains the largest collection of identities for elementary and special functions ever assembled. The site is generated from a set of *Mathematica* notebooks with typeset versions of all identities. The notebooks contain about 90,000 mathematical formulas. Because *Mathematica* notebooks are structured ASCII files that can be processed and manipulated programmatically by the *Mathematica* kernel, *Mathematica* can read and “understand” the formulas. Therefore, *Mathematica* can completely analyze and classify all the identities with respect to their mathematical structure and the functions that occur in them. The results of this analysis allow us to build a semantic search engine for mathematical identities. I will discuss the backend of the current mathematical search interface deployed on the Wolfram Functions site.

■ Introduction

In the issue 9:1 Corner I discussed various aspects of the Wolfram Functions site. I explained its organization, gave examples of identities, and showed sneak previews of the graphics gallery, which has since been added. Within an NSF grant with the Grainger Engineering Library of the University of Illinois at Urbana–Champaign and *MathWorld*TM, Oleg Marichev, John Renze, Chris Williamson, Andy Hunt, and I worked on various enhancements to the site over the last year. The new components are interactive plotting, the calculation of function values, and a mathematical search engine. In this Corner, I will discuss some of the implementation issues of the mathematical search. Contrary to my other Corners, this one will contain virtually no *Mathematica* code or graphics, but mostly text. Because this is the first operational semantic search engine on the World Wide Web, I believe a look behind the working mechanisms and a bird's eye view of the search strategy will be more interesting to most users than a variety of code snippets. Of course, the preparation of the data, as well as the individual mathematical searches, are realized through *Mathematica* programs.

■ Why Do We Need a Mathematical Search?

As emphasized in the penultimate Corner, the Wolfram Functions site can be viewed as a large table, with more than 250 functions running horizontally and more than 36 properties running vertically. Despite this clear organization, finding a special identity that you need (or vaguely remember) can be time-consuming. Some of the resulting 9,000 (250×36) matrix entries have up to five levels of nested subsections. (For instance, the entry for indefinite integrals of the Cos function contains thousands of entries.) Many identities can be written in different forms and classified in different ways. Furthermore, some identities might be given in a more general form than needed for a concrete purpose. So a search engine is clearly in order for quick and convenient access to the vast amount of knowledge encoded in the identities. Currently our Wolfram websites use a Google box to search textual content. Because all the pages with identities have the input form of the identities, you can already carry out some level of content-oriented searching. But for more complicated searches with a well-defined mathematical pattern in mind, a text search cannot give a satisfactory result. So we decided to implement a semantic search engine.

■ Hierarchical Menus versus *Mathematica* Patterns

How should a mathematical search be specified? On the one hand, we are all used to a Google-style search box that specifies words or phrases to occur or to not occur. But specifying a mathematical formula through text is not standardized. Too few people are fluent enough to specify MathML-based searches. In addition, many of the more complicated special functions are not immediately available in the MathML markup language. Similar remarks hold for $\text{T}_{\text{E}}\text{X}$ -based searches. *Mathematica* patterns are a natural way to specify semantic mathematics programmatically. While the deployed search page allows specifying a *Mathematica* pattern, even this turns out not to be optimal. While in principle one could specify any formula present on the Wolfram Functions site in this way, in practice there are two main disadvantages.

1. *Mathematica* patterns are primarily used for representing structural content. While for many functions there is a canonical isomorphism between the structure (say `Sin`) and the mathematical meaning (the function $\sin$), for more complicated expressions the two worlds are no longer isomorphic.

There are typically many structurally inequivalent ways to encode the same expression (`Sqrt[x]` versus `Power[x, 1/2]`, `Exp[x]` versus `E^x`, or `D` versus `Derivative`). Putting the burden of specifying all mathematically equivalent expressions on the searcher is inconvenient.

2. Specifying that a certain expression should (or should not) appear on one side of an equation, asymptotic expansion, or inequality leads to relatively large patterns.

To make searching convenient (without assuming prior knowledge of any computer language), we decided to construct a pull-down menu-driven interface. It allows users to specify functions that occur (or do not), constants, numbers, and operations on equations, which can appear on the left-hand side, right-hand side, or both sides. Functions are grouped according to the scheme already in use on the site, such as Elementary Functions, Bessel and Airy Functions, and so on.

Presently the following operations can be specified. About half of these operations are currently not represented as built-in functions in *Mathematica*.

- Differentiation • Series expansion • Indefinite integration • Definite integration • Summation • Product • Limit • Continued fractions • Singularities • Branch cuts • Branch points • Analyticity boundary • Discontinuity sets • Ramification indices • Wronskian • Fourier transform • Inverse Fourier transform • Fourier cos transform • Fourier sin transform • Laplace transform • Inverse Laplace transform • Mellin transform • Inverse Mellin transform • Hilbert transform • Hankel transform

■ Analyzing an Identity

Carrying out a mathematical search is possible because in *Mathematica* notebooks the typeset formulas are unique representations (modulo unimportant choices) of the mathematical meaning of the encoded identities. As a concrete example, here is the cell corresponding to the functional equation of the Riemann zeta function, identity 10.01.16.0001.

```
In[1]:= identityCell =
  Cell[BoxData[RowBox[{RowBox[{"Zeta", "[", "s", "]"}], "=",
    RowBox[{RowBox[{"Gamma", "[", RowBox[{"1", "-", "s"}], "]"}],
      SuperscriptBox["2", "s"], " ",
      SuperscriptBox["π", RowBox[{"s", "-", "1"}]], " ",
      RowBox[{"Sin", "[", FractionBox[RowBox[{"π"}, "2"], "s"}], " ",
        RowBox[{"Zeta", "[", RowBox[{"1", "-", "s"}], "]"}]}], (*for here*)"Output"];
```

Because notebooks themselves are *Mathematica* expressions, much the same as $\text{Sin}[x]$, we can treat documents programmatically. The formulas, identities, and equations contained in notebooks can be converted from their textual (box) representation to semantically meaningful *Mathematica* expressions.

Here is the formatted form of the cell.

```
In[2]:= CellPrint@identityCell

Zeta[s] == Gamma[1 - s] 2^s π^{s-1} Sin[π s / 2] Zeta[1 - s]
```

We interpret it and immediately wrap a `Hold` or `HoldForm` around the interpreted form to avoid any auto-evaluation, which might change the form of an identity or potentially take a long time for identities that contain integrals.

```
In[3]:= identity = ToExpression[#, StandardForm, HoldForm] &@identityCell[[1]]
```

```
Out[3]= Zeta[s] == Gamma[1 - s] 2^s Pi^{s-1} Sin[Pi s/2] Zeta[1 - s]
```

In the next example, we generate MathML, input form, gif versions, and the traditional form of this identity.

```
In[4]:= MathMLForm[identity] // ToString //
```

```
(* write in less verbose manner *)StringReplace[#, "\n" -> " "] & //  
FixedPoint[StringReplace[#, " " -> " "] &, #] &
```

```
Out[4]= <math><mrow><semantics><mrow><mi>&#950;</mi><mo>&#8289;</mo>  
<mo></mo></mi></mi><mo></mo></mrow><annotation  
encoding='Mathematica'>TagBox[RowBox[List[&quot; \[Zeta]&  
&quot;; , &quot;; (&quot;; , TagBox[&quot;s&quot;; , Zeta, Rule[Editable,  
True]], &quot;;)&quot;;]], InterpretTemplate[Function[BoxForm'e$,  
Zeta[BoxForm'e$]]]]</annotation></semantics><mo>&#63449;</mo>  
<mrow><mrow><mi>&#915;</mi><mo>&#8289;</mo><mo>  
(</mo><mrow><mn>1</mn><mo>-</mo><mi></mi></mrow>  
<mo></mo></mrow><mo>&#8290;</mo><msup><mn>  
2</mn><mi>s</mi></msup><mo>&#8290;</mo><msup><  
mi>&#960;</mi><mrow><mi>s</mi><mo>-</mo><mn>1</mn>  
</mrow></msup><mo>&#8290;</mo><mrow><mi>sin</mi>  
<mo>&#8289;</mo><mo></mo><mfrac><mrow><  
mi>&#960;</mi><mo>&#8290;</mo><mi>s</mi></mrow><  
mn>2</mn></mfrac><mo></mo></mrow><mo>&#8290;</mo>  
<mo><semantics><mrow><mi>&#950;</mi><mo>&#8289;</mo>  
<mo><mo></mo><mrow><mn>1</mn><mo>-</mo><mi>  
</mi></mrow><mo></mo></mrow><annotation  
encoding='Mathematica'>TagBox[RowBox[List[&quot; \[Zeta]&  
&quot;; , &quot;; (&quot;; , TagBox[RowBox[List[&quot;1&quot;; , &  
&quot;;-&quot;; , &quot;;s&quot;;]], Zeta, Rule[Editable, True]], &  
&quot;;)&quot;;]], InterpretTemplate[Function[BoxForm'e$, Zeta[  
BoxForm'e$]]]]</annotation></semantics></mrow></mrow></math>
```

```
In[5]:= InputForm[identity]
```

```
Out[5]//InputForm= HoldForm[Zeta[s] == Gamma[1 - s]*2^s*Pi^(s - 1)*Sin[(Pi*s)/2]*Zeta[1 -  
s]]
```

```
In[6]:= Show@ImportString[
```

```
ExportString[Cell[MakeBoxes[#, TraditionalForm] &@identity,  
"Output"], "GIF"], "GIF"]
```

$$\zeta(s) = \Gamma(1-s) 2^s \pi^{s-1} \sin\left(\frac{\pi s}{2}\right) \zeta(1-s)$$

Now we analyze the identity. It is an equality as opposed to an asymptotic expansion or an inequality.

```
In[7]:= Head@&#215;FunctionalEquation
```

```
Out[7]= Equal
```

Its fundamental building blocks are the following functions, numbers, and constants.

```
In[8]:= Union[Level[#, {-1}, Heads → True]] &@ξFunctionalEquation
```

```
Out[8]= {-1, 1/2, 1, 2, Equal, Gamma, π, Plus, Power, s, Sin, Times, Zeta}
```

The identity contains six different numerical functions.

```
In[9]:= Select[%,
  Head[#] == Symbol && MemberQ[Attributes[#], NumericFunction] &] &@
  ξFunctionalEquation
```

```
Out[9]= {Gamma, Plus, Power, Sin, Times, Zeta}
```

These are all the nontrivial subexpressions of the identity.

```
In[10]:= Union[Level[#, {1, -2}, Heads → True]] &@ξFunctionalEquation
```

```
Out[10]= {2^s, π^{-1+s}, 1-s, -1+s, -s, π s/2, Gamma[1-s], Sin[π s/2],
  Zeta[1-s], 2^s π^{-1+s} Gamma[1-s] Sin[π s/2] Zeta[1-s], Zeta[s]}
```

The production-quality analysis would continue by removing any dummy variables of summation or integration and by making the identities independent of variables that are not built in, such as s in the last example. Because many *Mathematica* functions come with a different number of arguments, they must be distinguished when analyzing an identity. (For example, the function *Zeta* is called with one argument in the case of the Riemann zeta function and with two arguments in the case of the Hurwitz zeta function.) Rational numbers are both kept intact as well as taken apart, so that their numerators and denominators can be considered as integers that appear in an identity. Then mathematically identical forms are created (like the power versus square root forms mentioned earlier). As a result, we have a detailed, multifaceted representation of each identity. In addition, we also store information about the section, subsection, ... where the identities have their natural place. Such information is used in the “Search for similar formulas” (see The Results Returned section).

■ Building Hash Tables

A more detailed version of the sample analysis just described is carried out on all 90,000 identities of the Wolfram Functions site. This one-time processing procedure takes a few hours. Then the connections *identity*→*listOfIngredients* are reversed and large hash tables of the form *ingredient*→*listOfIdentities* are constructed. While quite large, such tables allow for a very fast lookup, which is of constant time and independent of the length of the table. The ingredients are sorted into four categories: functions (such as *Sin*, *BesselJ*), constants (such as *Pi*, *E*), numbers (such as (2, 3, 1729), and operations (such as *Sum*, *Integrate*). Integrals are classified as definite versus indefinite.

Here are a few example counts.

Ingredient	Occurrences in the Left-Hand Side	Occurrences in the Right-Hand Side	Occurrences in Conditions
Cos	9517	6234	66
Pi	2566	42730	12237
Hypergeometric2F1	4088	1421	209
D	2372	358	39
11	596	211	19

■ Hypergeometric Functions

About one-sixth of all the identities on the site contain hypergeometric functions. Many of them occur in the general form ${}_pF_q$ for arbitrary positive integers p and q . This means that each of these identities encodes a whole family of identities for $\{p, q\} = \{0, 0\}, \{1, 0\}, \{0, 1\}, \{1, 1\}, \dots, \{6, 5\}, \dots$. In the building phase of the hash tables, the most common instances are explicitly generated and then analyzed.

It often happens (for instance as a result of **Integrate**) that you want to reduce a hypergeometric function to simpler functions like elementary functions, Bessel functions, error functions, and so on. The site contains thousands of such special cases. Because the first and second arguments can be interchanged and the elements in each argument group can be reordered, finding these specializations can be time-consuming. We wrote a special purpose search code for these cases. Given a pattern of a hypergeometric function, say ${}_2F_2(a, a+1; 1-a, b-1; z)$, the four possible pairs of realizations $\{a_1, a_2; b_1, b_2\} = \{a, a+1; 1-a, b-1\}, \{a+1, a; 1-a, b-1\}, \dots$ are constructed and matched against all structurally matching hypergeometric functions in the collection. For each such realization and each member of the collection, **Reduce** is called to find out if a match is possible. No explicit realizations are needed, but impossible matches must be ruled out. While this is a relatively time-consuming procedure (of the order of 10^1 seconds for a typical hypergeometric function), it finds all hypergeometric identities that match a certain pattern. See Search 7 for an example.

■ The General Search Process

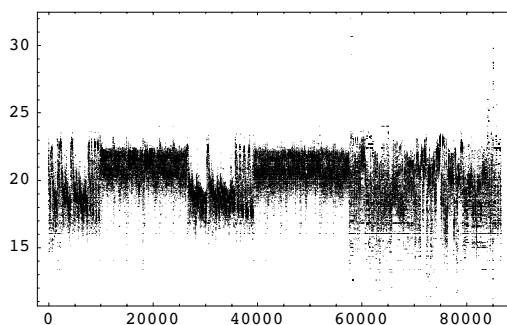
After parsing the objects selected in the pull-down menus (whether function, constant, number, or operation; that are either to occur or not), we form a *Mathematica* expression representing the search. To find the identities that match, we start with a list of all identities in the form of a list of strings of the identity numbers. Then for each object, a call is made to the prepared hash tables. The identity numbers returned are either joined or complemented with the current list (according to whether OR or AND was specified in the search). Such a look-up and reduction step is done consecutively for all the specified

search criteria, resulting in a list of identity numbers that match them all. Because the look-up operations and list manipulations are fast in *Mathematica*, even complicated searches typically take a fraction of a second of CPU time, including all preprocessing and postprocessing (with the exception of searches for mathematically matching hypergeometric functions).

If present in the search, *Mathematica* patterns are treated differently. First, they are put into canonical form (carefully avoiding any evaluation). Then, as much as possible without actually calling a function like `MatchQ`, the mathematical meaning is inferred from a structural pattern (for instance, the structural pattern `_ArcTan` encodes the two functions `ArcTan[z]` and `ArcTan[x, y]`) and used in a hash table lookup. More complicated patterns that contain pattern tests and conditions are wrapped with `HoldPattern` and matched literally against held versions of all identities.

If specified, filter options to return only those identities that contain basic arithmetic operations and either only elementary functions or only integer functions are applied to the result. To decide which functions are present in an identity, pregenerated hash tables are used, too.

Finally, the matching identities are sorted by complexity using leaf count, byte count, and the number of functions and variables; this is remotely similar to the default measure used for “simplicity” in *Simplify*. We use a linear combination of byte count and leaf count to compensate for any large atomic expressions such as large integers. While on average we have 1 *byte count* $\approx$ 22 *leaf counts*, many identities substantially deviate from this average. The following graphic compares leaf counts and byte counts for all identities from the Wolfram Functions site.



The complexities are precalculated for all identities, allowing for fast sorting.

A further possibility for sorting is to penalize extra functions. Suppose you make a search for all identities that contain the functions `Cos`, `Sin`, and `Tan`. Then identities that contain only these three functions (and basic arithmetic functions) would be returned first.

■ The web $Mathematica$ Interface

John Renze has implemented a convenient web interface that can easily be extended to include more search criteria. Pull-down menus specify which functions, constants, numbers, and operations should be included or not included in the search parameters. To avoid long pull-down menus, functions are grouped into Elementary Functions, Bessel and Airy Functions, and so on. Being able to freely add or remove search criteria is a nice feature typically found only in mail programs, not on web pages.

After clicking the Search button, a canonical form of the search request is sent to a server that runs web $Mathematica$. The server starts a $Mathematica$ session, the specified search is carried out, the results are analyzed, and a web page containing gifs of the resulting identities is assembled and returned.

■ The Results Returned

Renze also implemented the format of the results. All the formulas found are, by default, presented as gifs and have hyperlinks to the corresponding page of the Wolfram Functions site. The identities can be downloaded in `StandardForm` in notebooks or in `TraditionalForm` in pdf files.

Google, the de facto standard for searching today, has a Similar pages search button. Everyone would agree that $\sin'(x) = \cos(x)$ and $\cos'(x) = -\sin(x)$ are similar identities. This suggests that a working definition of “similar” would include all identities that contain other functions from the same group in semantically equivalent positions. We implemented a search for similar identities using a Hamming-type distance function. It treats functions from the same group as equivalent and adds a few rules to make operations such as differentiation and integration similar. To my surprise, the resulting search worked unexpectedly well. So, we continued refining the definition for “similar,” and the current functionality seems useful and natural.

■ Some Examples

After all the explanations about how search is implemented, it is time for some examples.

We start with a search for identities containing the function `Cos` and its inverse `Cos(-1)`. We restrict the search to elementary functions. Clicking on the following hyperlink will bring up the mathematical search page on the website with the corresponding fields filled in. [Search 1](#)

We obtain about 20 results with the first few shown here.

01.07.16.0005.01

$$\text{Cos}[\text{ArcCos}[z]] == z$$

01.07.21.0489.01

$$\int \cos[\operatorname{ArcCos}[z]] \, dz = \frac{z^2}{2}$$

01.13.17.0001.01

$$\cos[w[z_1] + w[z_2]]^2 - 2 z_1 z_2 \cos[w[z_1] + w[z_2]] + z_1^2 + z_2^2 = 1 / ; \\ w[z] = \operatorname{ArcCos}[z]$$

01.07.16.0018.01

$$\cos\left[\frac{\operatorname{ArcCos}[z]}{2}\right] = \frac{\sqrt{1+z}}{\sqrt{2}}$$

01.07.21.0490.01

$$\int \cos[a \operatorname{ArcCos}[z]] \, dz = \\ \frac{1}{2} \left(-\frac{\cos[(-1+a) \operatorname{ArcCos}[z]]}{-1+a} + \frac{\cos[(1+a) \operatorname{ArcCos}[z]]}{1+a} \right)$$

We continue with a search for all integral representations of Euler gamma, where γ must appear on the left-hand side and a definite integral must appear on the right-hand side. [Search 2](#)

We again find about 20 results. Here are some of them.

02.06.07.0002.01

$$\text{EulerGamma} = - \int_0^1 \log[-\log[t]] \, dt$$

02.06.07.0001.01

$$\text{EulerGamma} = - \int_0^\infty e^{-t} \log[t] \, dt$$

02.06.07.0014.01

$$\text{EulerGamma} = \sum_{k=1}^{n-1} \frac{1}{k} + \frac{1}{2n} - \log[n] + 2 \int_0^\infty \frac{t}{(t^2 + n^2)(e^{2\pi t} - 1)} \, dt$$

Next we search for limit representations of the Heaviside theta function (`UnitStep` in *Mathematica*). [Search 3](#)

Here are the first few of 15 formulas found.

14.01.09.0002.01

$$\text{UnitStep}[x] = \lim_{\varepsilon \rightarrow +0} \left[\exp\left[\exp\left[-\frac{x}{\varepsilon}\right]\right] \right]$$

14.01.09.0001.01

$$\text{UnitStep}[x] = \lim_{\varepsilon \rightarrow +0} \left[\frac{1}{1 + \exp\left[-\frac{x}{\varepsilon}\right]} \right]$$

14.01.09.0003.01

$$\text{UnitStep}[x] = \frac{1}{2} \left(\lim_{\varepsilon \rightarrow +0} \left[\tanh\left[\frac{x}{\varepsilon}\right] \right] + 1 \right)$$

14.01.09.0006.01

$$\text{UnitStep}[x] == \frac{1}{2} \left(\text{Limit} \left[\text{Erf} \left[\frac{x}{\varepsilon} \right], \varepsilon \rightarrow +0 \right] + 1 \right)$$

14.01.09.0007.01

$$\text{UnitStep}[x] == \frac{1}{2} \text{Limit} \left[\text{Erfc} \left[-\frac{x}{\varepsilon} \right], \varepsilon \rightarrow +0 \right]$$

Our next search will be for all formulas with elementary functions that contain the integers 1 to 9. [Search 4](#)

About 160 formulas are found. They are mainly products and integrals. Here are the first few.

02.05.08.0003.01

$$e == 2 \left(\frac{2}{1} \right)^{\frac{1}{2}} \left(\frac{2}{3} \frac{4}{3} \right)^{\frac{1}{4}} \left(\frac{4}{5} \frac{6}{5} \frac{6}{7} \frac{8}{7} \right)^{\frac{1}{8}} \left(\frac{8}{9} \frac{10}{9} \frac{10}{11} \frac{12}{11} \frac{12}{13} \frac{14}{13} \frac{14}{15} \frac{16}{15} \right)^{\frac{1}{16}} \dots$$

01.03.21.0117.01

$$\int z^4 e^{a\sqrt{z}} dz == \frac{1}{a^{10}} \left(2 e^{a\sqrt{z}} \left(-362880 + 362880 a \sqrt{z} - 181440 a^2 z + 60480 a^3 z^{3/2} - 15120 a^4 z^2 + 3024 a^5 z^{5/2} - 504 a^6 z^3 + 72 a^7 z^{7/2} - 9 a^8 z^4 + a^9 z^{9/2} \right) \right)$$

01.24.21.0472.01

$$\int \frac{-3 \tanh[z] + (\text{Sech}[z]^6 \tanh[z])^{1/3}}{(\cosh[z]^5 \sinh[z])^{2/3}} dz == \left(3 \sinh[z] \left(-72 (15 + 12 \text{Sech}[z]^2 + 8 \text{Sech}[z]^4) \tanh[z]^2 + 252 (7 + 3 \cosh[2z]) \sinh[z]^2 (\text{Sech}[z]^6 \tanh[z])^{2/3} + 5 \cosh[z]^2 (55 + 48 \cosh[2z] + 9 \cosh[4z]) (\text{Sech}[z]^6 \tanh[z])^{4/3} \right) \right) / \left(1120 (\cosh[z]^5 \sinh[z])^{2/3} (\text{Sech}[z]^6 \tanh[z])^{2/3} (-3 \sinh[z] + \cosh[z] (\text{Sech}[z]^6 \tanh[z])^{1/3}) \right)$$

To find the functional equation of the Riemann zeta function, we search for all identities that contain the Riemann zeta on both sides of the equation. [Search 5](#)

We find 13 identities and show the first few. The first establishes a general symmetry, the second that Zeta has its own asymptotics as infinity, and the third is the classic functional equation. Further equations contain finite and infinite sums of zeta functions.

10.01.04.0002.01

$$\text{Zeta}[\text{Conjugate}[s]] == \text{Conjugate}[\text{Zeta}[s]]$$

10.01.06.0006.01

$$\text{Zeta}[s] \propto \text{Zeta}[s] / ; (\text{Abs}[s] \rightarrow \infty)$$

10.01.16.0001.01

$$\text{Zeta}[1-s] == \frac{\pi^{\frac{1}{2}-s} \text{Gamma}\left[\frac{s}{2}\right]}{\text{Gamma}\left[\frac{1-s}{2}\right]} \text{Zeta}[s]$$

10.01.17.0001.01

$$\text{Zeta}[s] == \frac{\pi^{-\frac{1}{2}+s} \text{Gamma}\left[\frac{1-s}{2}\right]}{\text{Gamma}\left[\frac{s}{2}\right]} \text{Zeta}[1-s]$$

10.01.17.0005.01

$$\sum_{k=1}^{n-1} \text{Zeta}[2k] \text{Zeta}[2n-2k] == \left(n + \frac{1}{2}\right) \text{Zeta}[2n] \quad ; n \in \text{Integers} \wedge n > 1$$

10.01.23.0002.01

$$\sum_{k=1}^{\infty} \frac{\text{Pochhammer}[s, k]}{k! 2^k} \text{Zeta}[k+s] == (2^s - 2) \text{Zeta}[s]$$

The organization of the Wolfram Functions site lets you easily browse through similar identities for one function. Through the search it is easy to find “equivalent” formulas for groups of functions. This search is for all continued fraction expansions of the inverse trigonometric functions. [Search 6](#)

Eleven such expansions are found. Here are some of them.

01.16.10.0004.01

$$\text{ArcCot}[z] == \frac{1}{z + \text{ContinueFraction}\left[\left\{1, \frac{(1+2k)z}{k}\right\}, \{k, 1, \infty\}\right]} \quad ;$$

$$\text{Not}[\text{IntervalMemberQ}[\text{Interval}[\{-1, 1\}], i z]]$$

01.14.10.0002.01

$$\text{ArcTan}[z] == \frac{z}{1 + \text{ContinueFraction}\left[\{k^2 z^2, 2k+1\}, \{k, 1, \infty\}\right]} \quad ;$$

$$\text{Not}[\text{IntervalMemberQ}[\text{Interval}[\{-\infty, -1\}], i z]] \wedge$$

$$\text{Not}[\text{IntervalMemberQ}[\text{Interval}[\{1, \infty\}], i z]]$$

01.17.10.0002.01

$$\text{ArcCsc}[z] ==$$

$$z^{-1} \sqrt{1-z^{-2}} \Big/ \left(1 + \text{ContinueFraction}\left[\left\{-2\left(2\text{Floor}\left[\frac{k+1}{2}\right] - 1\right)\right.\right.\right.$$

$$\left.\left.\text{Floor}\left[\frac{k+1}{2}\right] z^{-2}, 2k+1\right\}, \{k, 1, \infty\}\right] \right) \quad ;$$

$$\text{Not}[\text{IntervalMemberQ}[\text{Interval}[\{-1, 1\}], z]]$$

01.12.10.0002.01

$$\text{ArcSin}[z] == z \sqrt{1-z^2} \Big/ \left(1 + \text{ContinueFraction}\left[\left\{-2\left(2\text{Floor}\left[\frac{k+1}{2}\right] - 1\right)\text{Floor}\left[\frac{k+1}{2}\right] z^2, 2k+1\right\}, \{k, 1, \infty\}\right] \right) \quad ;$$

$$\text{Not}[\text{IntervalMemberQ}[\text{Interval}[\{-\infty, -1\}], z]] \wedge$$

$$\text{Not}[\text{IntervalMemberQ}[\text{Interval}[\{1, \infty\}], z]]$$

01.18.10.0002.01

$$\begin{aligned} \text{ArcSec}[z] = & \frac{\pi}{2} - z^{-1} \sqrt{1 - z^{-2}} \Big/ \left(1 + \text{ContinueFraction} \left[\left\{ -2 \left(2 \text{Floor} \left[\frac{k+1}{2} \right] - 1 \right) \right. \right. \right. \\ & \left. \left. \left. \text{Floor} \left[\frac{k+1}{2} \right] z^{-2}, 2k+1 \right\}, \{k, 1, \infty\} \right] \right) /; \\ & \text{Not}[\text{IntervalMemberQ}[\text{Interval}[\{-1, 1\}], z]] \end{aligned}$$

01.13.10.0002.01

$$\begin{aligned} \text{ArcCos}[z] = & \frac{\pi}{2} - z \sqrt{1 - z^2} \Big/ \left(1 + \text{ContinueFraction} \left[\left\{ -2 \left(2 \text{Floor} \left[\frac{k+1}{2} \right] - 1 \right) \right. \right. \right. \\ & \left. \left. \left. \text{Floor} \left[\frac{k+1}{2} \right] z^2, 2k+1 \right\}, \{k, 1, \infty\} \right] \right) /; \\ & \text{Not}[\text{IntervalMemberQ}[\text{Interval}[\{-\infty, -1\}], z]] \wedge \\ & \text{Not}[\text{IntervalMemberQ}[\text{Interval}[\{1, \infty\}], z]] \end{aligned}$$

Our next search contains hypergeometric functions. We will find all identities that contain ${}_3F_2(a, a+1, b-1; a, b+1; z)$ mathematically. [Search 7](#)

Because (as discussed earlier) many thousand potential realizations must be tested, this search will take a few seconds. About 25 matches are found and here are three of them. The formulas returned have arguments that are consistent with those given in the original hypergeometric function.

07.27.03.0116.01

$$\begin{aligned} \text{HypergeometricPFQ}[\{a, b, c\}, \{d, c\}, z] = & \\ \text{Hypergeometric2F1}[a, b, d, z] & \end{aligned}$$

07.27.03.0039.01

$$\begin{aligned} \text{HypergeometricPFQ}[\{-n, b, c\}, \{b-1, c-m\}, 1] = & 0 /; n \in \text{Integers} \wedge \\ n \geq 0 \wedge m \in \text{Integers} \wedge m \geq 0 \wedge l \in \text{Integers} \wedge l > 0 \wedge 1-l \leq m \leq n-1-l \end{aligned}$$

07.27.03.0010.01

$$\begin{aligned} \text{HypergeometricPFQ}[\{a, b, c\}, \{a-n, b+m\}, 1] = & \\ (\text{Gamma}[1-c] \text{Gamma}[b+m] \text{Pochhammer}[1+b-a, n]) / & \\ (\text{Gamma}[b-c+1] (m-1)! \text{Pochhammer}[1-a, n]) & \\ \sum_{k=0}^{m-1} (\text{Pochhammer}[1+b-a+n, k] \text{Pochhammer}[b, k] \text{Pochhammer}[1-m, & \\ k]) / (k! \text{Pochhammer}[1+b-a, k] \text{Pochhammer}[1+b-c, k]) /; & \\ \text{Re}[c] < m-n \wedge m \in \text{Integers} \wedge m \geq 0 \wedge n \in \text{Integers} \wedge n \geq 0 & \end{aligned}$$

Our final direct search will be for derivatives of the Bessel function $\mathcal{J}$. This means that we want the $\mathcal{J}$ function and differentiation on the left-hand side of an identity. [Search 8](#)

One of the first matches returned is the following formula for the second derivative of $\mathcal{J}_\nu(z)$.

03.01.20.0010.01

$$\partial_{\{z,2\}} \text{BesselJ}[\nu, z] == \frac{1}{4} (\text{BesselJ}[\nu - 2, z] - 2 \text{BesselJ}[\nu, z] + \text{BesselJ}[2 + \nu, z])$$

Clicking the Search for similar formulas button results in about 30 matches. The similarity of the formulas returned consists in either equivalent formulas for the other three Bessel functions, first-order derivatives, or integrals containing simple Bessel functions. Here are the first seven similar formulas. [Search 9](#)

03.01.20.0006.01

$$\partial_z \text{BesselJ}[\nu, z] == \frac{1}{2} (\text{BesselJ}[\nu - 1, z] - \text{BesselJ}[\nu + 1, z])$$

03.03.20.0010.01

$$\partial_{\{z,2\}} \text{BesselY}[\nu, z] == \frac{1}{4} (\text{BesselY}[-2 + \nu, z] - 2 \text{BesselY}[\nu, z] + \text{BesselY}[2 + \nu, z])$$

03.02.20.0010.01

$$\partial_{\{z,2\}} \text{BesselI}[\nu, z] == \frac{1}{4} (\text{BesselI}[-2 + \nu, z] + 2 \text{BesselI}[\nu, z] + \text{BesselI}[2 + \nu, z])$$

03.04.20.0010.01

$$\partial_{\{z,2\}} \text{BesselK}[\nu, z] == \frac{1}{4} (\text{BesselK}[-2 + \nu, z] + 2 \text{BesselK}[\nu, z] + \text{BesselK}[2 + \nu, z])$$

03.03.20.0006.01

$$\partial_z \text{BesselY}[\nu, z] == \frac{1}{2} (\text{BesselY}[\nu - 1, z] - \text{BesselY}[\nu + 1, z])$$

03.04.20.0006.01

$$\partial_z \text{BesselK}[\nu, z] == -\frac{1}{2} (\text{BesselK}[\nu - 1, z] + \text{BesselK}[\nu + 1, z])$$

03.02.20.0006.01

$$\partial_z \text{BesselI}[\nu, z] == \frac{1}{2} (\text{BesselI}[\nu - 1, z] + \text{BesselI}[\nu + 1, z])$$

■ Conclusions

The Wolfram Functions site contains a large body of mathematical knowledge that can be read and understood by both humans and computers. Because we can give semantic meaning to a typeset formula within *Mathematica*, it is possible to build a truly semantic search engine as a *Mathematica* program running through a web*Mathematica* interface for the world of special functions, a self-contained part of mathematics.

Most people use the identities of the Wolfram Functions site to do *Mathematica* calculations. As a result, further plans include the possibility of running the

search from *Mathematica* through web services and returning the identities found as immediately usable rules (with head `RuleDelayed` instead of `Equal`). We will also soon add introductory text for many functions and function groups, and we will begin adding tables of zeros and related tabular data.

Any comments about functionality or additions to the identity collection of the Wolfram Functions site are always welcome.

Michael Trott

Special Functions Developer
Wolfram Research, Inc.
mtrott@wolfram.com

Moment-Based Density Approximants

Serge B. Provost

It is often the case that the exact moments of a statistic of the continuous type can be explicitly determined, while its density function either does not lend itself to numerical evaluation or proves to be mathematically intractable. The density approximants discussed in this article are based on the first n exact moments of the corresponding distributions. A unified semiparametric approach to density approximation is introduced. Then, it is shown that the resulting approximants are mathematically equivalent to those obtained by making use of certain orthogonal polynomials, such as the Legendre, Laguerre, Jacobi, and Hermite polynomials. Several examples illustrate the proposed methodology.

■ 1. Introduction

This article is concerned with the problem of approximating a density function from the moments (or cumulants) of a given distribution. Approximants of this type can be obtained, for example, by making use of Pearson or Johnson curves, see [1, 2, 3], or saddlepoint approximations as discussed in [4]. These methodologies can provide adequate approximations in a variety of applications involving unimodal distributions. However, they may prove difficult to implement and their applicability can be subject to restrictive conditions. The approximants proposed in this article are expressed in terms of relatively simple formulae and apply to a very wide array of distributions. Moreover, their accuracy can be improved by use of additional moments. Interestingly, another technique called the inverse Mellin transform, which is based on the complex moments of certain distributions, provides representations of their exact density functions in terms of generalized hypergeometric functions; for theoretical considerations as well as various applications, see [5, 6].

First, it should be noted that the b th moment of a statistic, $u(X_1, \dots, X_n)$, whose exact density is unknown, can be determined exactly or numerically by integrating the product $u(x_1, \dots, x_n)^b g(x_1, \dots, x_n)$ over the range of integration of the x_i 's, where $g(x_1, \dots, x_n)$ denotes the joint density of the X_i 's, $i = 1, 2, \dots, n$; for instance, this approach is used in Example 1. Alternatively, the moments of a random variable X can be obtained from the derivatives of its moment-generating function as is done in Example 4 or by making use of a relationship between the moments and the cumulants when the latter are known [7]. Moments can

also be derived recursively as is the case, for instance, in connection with certain queueing models. When the moments of a statistic uniquely determine its distribution and a sufficient number of moments are known, we can often approximate its density function in terms of sums involving orthogonal polynomials of a certain type. Conveniently, such polynomials are available as built-in *Mathematica* functions.

Density approximants based on Legendre and Laguerre polynomials are discussed in Sections 2 and 3, respectively, for random variables having finite and semi-infinite supports. The main formulae which allows for the *direct* evaluation of the density approximants are equations (15), derived in Section 2, and (29), obtained in Section 3. The approximant that is expressed in terms of Laguerre polynomials applies to a wide class of statistics which includes those whose asymptotic distribution is chi-square, such as $-2 \ln \lambda$, where λ denotes a likelihood ratio statistic, as well as those that are distributed as quadratic forms in normal variables, such as the sample serial covariance. It should be noted that an indefinite quadratic form can be expressed as the difference of two independent nonnegative definite quadratic forms whose cumulants are well known. As for distributions having compact supports, there are, for example, the Durbin-Watson statistic, Wilks' likelihood ratio criterion, the sample correlation coefficient, as well as many other useful statistics that can be expressed as the ratio of two quadratic forms, as discussed in [8].

In Section 4, we propose a unified density estimation methodology which only requires the moments of the distribution to be approximated and those of a suitable 'base density function'. As it turns out, this approach yields density approximants that are identical to those obtained from certain orthogonal polynomials—namely, the Legendre, Laguerre, Jacobi, and Hermite polynomials—whose associated weight function is proportional to the corresponding base density function. Several examples illustrate the various results. The *Mathematica* code utilized for implementing the main formulae and plotting the graphs is supplied in the Appendix.

For results in connection with the convergence of approximating sums that are expressed in terms of orthogonal polynomials, see [9, 10, 11, 12]. Since the proposed methodology allows for the use of a large number of theoretical moments and the functions being approximated are nonnegative, the approximants can be regarded as nearly exact *bona fide* density functions, and quantiles can thereupon easily be estimated with great accuracy. As well, the representations of the approximants make them easy to report and amenable to complex calculations.

Until now, orthogonal polynomials have been scarcely discussed in the statistical literature in connection with the approximation of distributions. This might have been due to difficulties encountered in deriving moments of high orders or in obtaining accurate results from high-degree polynomials. In any case, given the powerful computational resources that are widely available these days, such complications can hardly be viewed as impediments any longer. It should be pointed out that the simple semiparametric technique proposed in Section 4 eliminates some of the complications associated with the use of orthogonal

polynomials while yielding identical density approximants. This article is self-contained, and the results presented herein potentially have a host of applications. Being that the subject matter of this article is density approximation as opposed to density estimation, it ought to be emphasized that the techniques presented herein are meant to be used in conjunction with exact moments rather than sample moments.

■ 2. Approximants Based on Legendre Polynomials

A polynomial density approximation formula which applies to any continuous distribution having a compact support is obtained in this section. This approximant is derived from an analytical result stated in [10], which is couched in statistical nomenclature in this section.

The density function of a continuous random variable X that is defined on the interval $[-1, 1]$ can be expressed as follows:

$$f_X[x] = \sum_{k=0}^{\infty} \lambda_k P_k[x], \quad (1)$$

where $P_k[x]$ is a Legendre polynomial of degree k in x , that is,

$$P_k[x] = \frac{1}{2^k k!} \frac{\partial^k}{\partial x^k} (x^2 - 1)^k = \sum_{i=0}^{\text{Floor}[k/2]} (-1)^i 2^{-k} \frac{(2k - 2i)!}{i! (k - i)! (k - 2i)!} x^{k-2i}, \quad (2)$$

$\text{Floor}[k/2]$ denoting the largest integer less than or equal to $k/2$, and

$$\lambda_k = \frac{2k + 1}{2} \sum_{i=0}^{\text{Floor}[k/2]} (-1)^i 2^{-k} \frac{(2k - 2i)!}{i! (k - i)! (k - 2i)!} \mu_X[k - 2i] = \frac{2k + 1}{2} P_k^*[\omega] \quad (3)$$

with $P_k^*[X] = P_k[X]$, wherein X^{k-2i} is replaced by the $(k - 2i)$ th moment of X :

$$\mu_X[k - 2i] = E(X^{k-2i}) = \int_{-1}^1 x^{k-2i} f_X[x] dx, \quad (4)$$

[13]. Legendre polynomials can also be obtained by means of a recurrence relationship, which is derived for instance in [9, 178]. Given the first n moments of X , $\mu_X[1], \dots, \mu_X[n]$, and setting $\mu_X[0] = 1$, the following truncated series denoted by $f_{X_n}[x]$ can be used as a polynomial approximation to $f_X[x]$:

$$f_{X_n}[x] = \sum_{k=0}^n \lambda_k P_k[x], \quad (5)$$

that is,

$$f_{X_n}[x] := \sum_{k=0}^n \left(\frac{2k+1}{2} \text{LegendreP}[k, X] /. X^{j-} \rightarrow \mu_X[j] \right) \text{LegendreP}[k, x] \quad (6)$$

in *Mathematica* notation, where the pattern matching symbol $\rightarrow$ (which is typed $\Rightarrow$ in Input mode) conveniently replaces each occurrence of X^j in $\text{LegendreP}[k, X]$ with $\mu_X[j]$. It should be noted that expressions involving $:=$ (excluding the punctuation at the end of the formulae) can readily be used in a *Mathematica* notebook.

As explained in [14, 439], this polynomial turns out to be the least-squares approximating polynomial of degree n that minimizes the integrated squared error, that is, $\int_{-1}^1 (f_X[x] - f_{X_n}[x])^2 dx$. As stated in [15, 106], the moments of any continuous random variable whose support is a closed interval uniquely determine its distribution. Moreover, as shown by [10, 304], the rate of convergence of the supremum of the absolute error, $|f_X[x] - f_{X_n}[x]|$, depends on $f_X[x]$ and n via a continuity modulus. It follows that more accurate approximants can always be obtained by making use of higher degree polynomials.

We now turn our attention to the more general case of a continuous random variable Y which is defined on the closed interval $[a, b]$. We denote its density function by $f_Y(y)$ and its k th moment by

$$\mu_Y[k] = E(Y^k) = \int_a^b y^k f_Y[y] dy, \quad k = 0, 1, \dots \quad (7)$$

As pointed out in the Introduction, alternative methods are available for evaluating the moments of a distribution when the exact density is unknown. On mapping Y onto X by means of the linear transformation

$$X = \frac{2Y - (a+b)}{b-a}, \quad (8)$$

we obtain the desired range for X , that is, the interval $[-1, 1]$. The j th moment of X , which is obtained as the expected value of the binomial expansion of $((2Y - (a+b))/(b-a))^j$ is given by

$$\mu_X[j] = \frac{1}{(b-a)^j} \sum_{k=0}^j \binom{j}{k} 2^k \mu_Y[k] (-1)^{j-k} (a+b)^{j-k}, \quad (9)$$

that is,

$$\mu_X[j_] := \text{Expand} \left[\left(\frac{2Y - (a+b)}{b-a} \right)^j \right] /. \text{Table}[Y^k \rightarrow \mu_Y[k], \{k, 0, j\}] \quad (10)$$

or equivalently

$$\mu_X[j_] := \text{Expand} \left[\left(\frac{2Y - (a+b)}{b-a} \right)^j \right] /. Y^{j-} \rightarrow \mu_Y[j]. \quad (11)$$

Equation (6) can then be used to provide an approximant to the density function of X . On transforming X back to Y with the affine change of variable specified in equation (8) and noting that $\partial X / \partial Y = 2 / (b - a)$, we obtain the following approximate density function for Y :

$$f_{Y_n}[y] = \frac{2}{b-a} \sum_{k=0}^n \lambda_k P_k \left[\frac{2y - (a+b)}{b-a} \right], \quad (12)$$

that is,

$$f_{Y_n}[y] := \sum_{k=0}^n \left(\frac{2k+1}{b-a} \text{LegendreP}[k, X] /. X^{j-} \rightarrow \mu_X[j] \right) \text{LegendreP} \left[k, \frac{2y - (a+b)}{b-a} \right] \quad (13)$$

in *Mathematica* notation. On combining equations (9) and (13), one obtains the following compact representation of the density approximant:

$$f_{Y_n}[y] := \sum_{k=0}^n \frac{2k+1}{b-a} \left(\text{LegendreP}[k, X] /. X^{j-} \rightarrow \sum_{k=0}^j \frac{j! 2^k \mu_Y[k] (-a-b)^{j-k}}{(b-a)^j k! (j-k)!} \right) * \text{LegendreP} \left[k, \frac{2y - (a+b)}{b-a} \right]. \quad (14)$$

Now, observing that $\text{LegendreP}[k, X]$ with X^j replaced by $\mu_X[j]$ as given earlier, is equivalent to $\text{Legendre}[k, (2Y - (a+b))/(b-a)]$ with Y^j replaced by $\mu_Y[j]$, we also obtain

$$f_{Y_n}[y] := \sum_{k=0}^n \frac{2k+1}{b-a} \left(\text{LegendreP} \left[k, \frac{2Y - (a+b)}{b-a} \right] /. Y^{j-} \rightarrow \mu_Y[j] \right) \text{LegendreP} \left[k, \frac{2y - (a+b)}{b-a} \right]. \quad (15)$$

Thus, given $\mu_Y[k]$, $k = 1, 2, \dots, n$, the first n moments of a random variable defined on the interval $[a, b]$, an n th-degree polynomial approximation of its density function can be directly obtained from equations (14) or (15).

It should be noted that the density approximants so obtained may be negative on certain subranges of the support of their distributions having low density. This will likely occur if an insufficient number of moments are being used. However, by mere inspection of the approximate density plot, we should be able to determine whether a higher degree polynomial ought to be used. Indeed, owing to the convergence of the approximant, the density function will converge everywhere to a nonnegative number as more moments are being used. If we wish to obtain a truly *bona fide* density function, we could always take a normalized function, $\zeta_{Y_n}[\cdot]$, which is initially defined as being equal to $f_{Y_n}[\cdot]$ except on subintervals where the latter is negative, wherein it is set equal to zero.

In the following application, a polynomial approximation is obtained for the density of V , the square of the distance between two points that are randomly distributed in the unit cube.

□ Example 1: Exact and Approximate Density Functions of V

Let $\mathbf{X} = (X_1, X_2, X_3)$ and $\mathbf{Y} = (Y_1, Y_2, Y_3)$ be two points in the unit cube whose coordinates, X_i and Y_i , $i = 1, 2, 3$, are all independently and uniformly distributed in the interval $[0, 1]$, and let V denote the square of the distance between these two random points, that is, $(Y_1 - X_1)^2 + (Y_2 - X_2)^2 + (Y_3 - X_3)^2$, whose support is the interval $[0, 3]$.

The closed-form representation of the density function of V that follows, which is believed to be original, was derived from the integral representations obtained by [16, Section 2.6.4] by making use of certain trigonometric identities as well as some of *Mathematica*'s algebraic simplification routines:

$$f_V(v) = d(v) I_{[0,1]}(v) + g(v) I_{[1,2]}(v) + s(v) I_{[2,3]}(v), \quad (16)$$

where the functions $d(v)$, $g(v)$, and $s(v)$ can be easily identified from the expression given in the Appendix for $f_V(v)$, and $I_A(v)$ denotes the indicator function which is equal to one whenever v belongs to the set A and zero otherwise.

The b th moment of V can be evaluated by integrating $v^b f_V(v)$ whenever $b \geq -1/2$. Alternatively, if the density function of V were not known, we could determine its b th moment from the following integral representation:

$$\int_0^1 \int_0^1 \int_0^1 \int_0^1 \int_0^1 \int_0^1 ((y_1 - x_1)^2 + (y_2 - x_2)^2 + (y_3 - x_3)^2)^b dx_1 dx_2 dx_3 dy_1 dy_2 dy_3, \quad (17)$$

whose evaluation can be handled by *Mathematica*. However, on noting that the density function of $\delta_j = (y_j - x_j)^2$ is $((1/\sqrt{\delta_j}) - 1) I_{[0,1]}(\delta_j)$, it is much more efficient to compute the b th moment of V as follows:

$$\int_0^1 \int_0^1 \int_0^1 (\delta_1 + \delta_2 + \delta_3)^b (\delta_1^{-1/2} - 1) (\delta_2^{-1/2} - 1) (\delta_3^{-1/2} - 1) d\delta_1 d\delta_2 d\delta_3. \quad (18)$$

Figure 1 shows the exact probability density function (PDF) of V obtained from equation (16) (solid line) superimposed on its thirteenth-degree polynomial approximation (dashed line) evaluated from equation (15) (or equivalently equation (13)) with $a = 0$ and $b = 3$. The exact and approximate cumulative distribution functions (CDFs) which can easily be evaluated by integration, appear in Figure 2, and their difference is plotted in Figure 3. The code that was used for plotting the graphs and evaluating the various functions is provided in the Appendix. In general, note that in order to avoid round-off errors, it is advisable to carry out the calculations with rational numbers. In this example, the moments already are in rational form. When this is not the case, the command `Rationalize[.]` can be used to obtain rational representations.

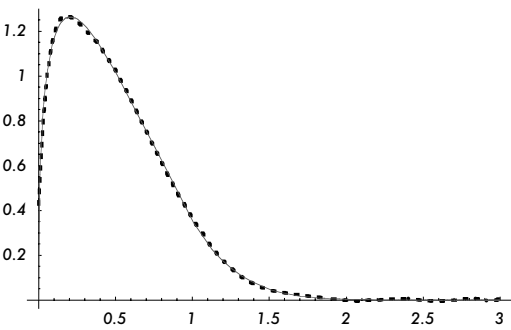


Figure 1. Exact and approximate (dashed line) PDFs. [Pq or Pq1 in the Appendix]

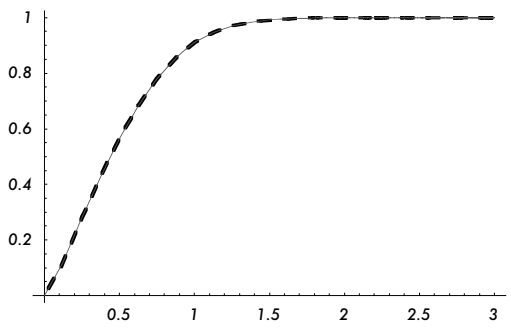


Figure 2. Exact and approximate (dashed line) CDFs. [PQ in the Appendix]

As Figure 3 indicates, the exact and approximate CDFs differ by less than 0.001 over the interval $[0, 3]$ in this case.

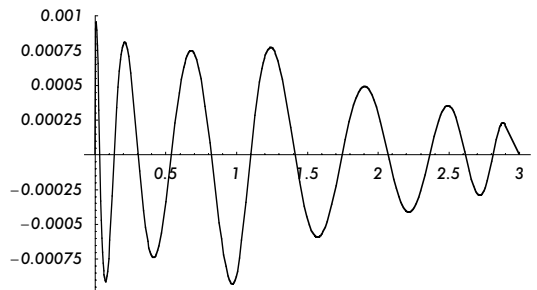


Figure 3. The difference between the exact and approximate CDFs. [Qd in the Appendix]

□ Example 2: Approximate Density of a Mixture of Beta Random Variables

Consider a mixture of two equally weighted beta distributions with parameters $(3, 2)$ and $(2, 30)$, respectively. A fifteenth-degree polynomial approximation was obtained from the compact formula given in equation (14). The exact density function of this mixture and its approximant, both plotted in Figure 4, are manifestly in close agreement. A glance at the *Mathematica* code that is provided in the Appendix for this example should convince the reader that very little programming is indeed required. Clearly, methodologies that are based on only a few moments would fail to provide satisfactory approximations in this case.

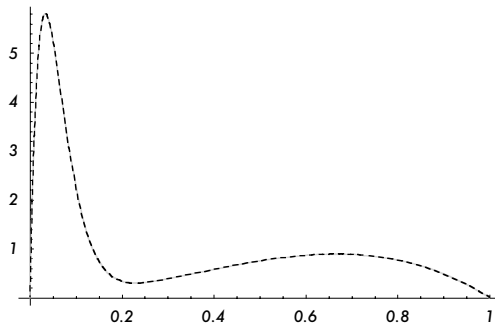


Figure 4. Exact and approximate (dashed line) PDFs. [Pb in the Appendix]

As specified in Section 4, approximants that are expressed in terms of Jacobi polynomials are ideally suited for approximating beta-type density functions. However, in the absence of prior knowledge about the shape of a density function, it is indicated to make use of approximants based on Legendre polynomials as they can theoretically accommodate any continuous distribution defined on a closed interval. It should be pointed out that if a density function turns out to be very irregular, a prohibitive number of moments might be required to approximate it satisfactorily. Thankfully, the majority of continuous distributions of interest are smooth and possess at most a few modes.

■ 3. Approximants Based on Laguerre Polynomials

As pointed out in the Introduction, the density functions of numerous statistics distributed on the positive half-line can be approximated from their exact moments by means of sums involving Laguerre polynomials. It should be pointed out that such approximants should only be used when the underlying distribution possesses the tail behaviour of a gamma random variable. Fortunately, this is often the case for test statistics whose support is semi-infinite. Note that for other types of distributions defined on the positive half-line, such as the

lognormal which is considered in Example 3, the moments may not uniquely determine the distribution; see [15, 106] for conditions ensuring that they do.

Consider a random variable Y defined on the interval $[a, \infty)$, whose j th moment is denoted by $\mu_Y[j]$, $j = 0, 1, 2, \dots$, and let

$$c = \frac{\mu_Y[2] - \mu_Y[1]^2}{\mu_Y[1] - a}, \quad (19)$$

$$v = \frac{\mu_Y[1] - a}{c} - 1, \quad (20)$$

and

$$X = \frac{Y - a}{c}. \quad (21)$$

As explained in Remark 3.1, when the parameters c and v are so chosen, the leading term of the approximant is a shifted gamma density function whose mean and variance agree with those of Y . Although a can be any finite real number, it is in most cases of interest equal to zero. By definition, c belongs to $\mathbb{R}^+$, the set of positive real numbers. Denoting the j th moment of X by

$$\mu_X[j] = E\left[\left(\frac{Y - a}{c}\right)^j\right], \quad (22)$$

that is,

$$\mu_X[j] := \sum_{b=0}^j \text{Expand}\left[\left(\frac{Y - a}{c}\right)^j\right] /. Y^{\text{h.}} \rightarrow \mu_Y[b], \quad (23)$$

the density function of the random variable X defined on the interval $[0, \infty)$ can be expressed as

$$f_X[x] = x^v e^{-x} \sum_{j=0}^{\infty} \delta_j L_j(v, x), \quad (24)$$

where

$$L_j[v, x] = \sum_{k=0}^j (-1)^k \frac{\Gamma(v + j + 1) x^{j-k}}{k! (j - k)! \Gamma(v + j - k + 1)} \quad (25)$$

is a Laguerre polynomial of order j in x with parameter v , that is, $\text{LaguerreL}[j, v, x]$ in *Mathematica* notation and

$$\delta_j = \sum_{k=0}^j (-1)^k \frac{j!}{k! (j - k)! \Gamma(v + j - k + 1)} \mu_X[j - k], \quad (26)$$

which also can be represented by $j! / \Gamma(v + j + 1)$ times $L_j[v, X]$, wherein X^k is replaced with $\mu_X[k]$, [13, 17]. Then, on truncating the series given in

equation (24) and making the change of variable $Y = cX + a$, we obtain the following density approximant for Y :

$$f_{Y_n}[y] = \frac{(y-a)^v e^{-(y-a)/c}}{c^{v+1}} \sum_{j=0}^n \delta_j L_j\left(v, \frac{y-a}{c}\right), \quad (27)$$

that is,

$$f_{Y_n}[y] := \frac{(y-a)^v e^{-(y-a)/c}}{c^{v+1}} \sum_{j=0}^n \frac{j!}{\text{Gamma}[v+j+1]} * \\ (\text{LaguerreL}[j, v, X] /. X^{k_{-}} \rightarrow \mu_X[k]) \text{LaguerreL}\left[j, v, \frac{y-a}{c}\right], \quad (28)$$

where $\text{Gamma}[\cdot] \equiv \Gamma[\cdot]$ denotes the gamma function or, observing that $\text{LaguerreL}[j, v, X]$ with X^k replaced by $\mu_X[k]$ as defined earlier, is equivalent to $\text{LaguerreL}[j, v, (Y-a)/c]$ with Y^k replaced by $\mu_Y[k]$, we obtain

$$f_{Y_n}[y] := \frac{(y-a)^v e^{-(y-a)/c}}{c^{v+1}} \sum_{j=0}^n \frac{j!}{(v+j)!} * \\ \left(\text{LaguerreL}\left[j, v, \frac{Y-a}{c}\right] /. Y^{k_{-}} \rightarrow \mu_Y[k] \right) * \text{LaguerreL}\left[j, v, \frac{y-a}{c}\right], \quad (29)$$

where c and v are defined in equations (19) and (20), respectively. It should be noted that the representation of the approximant appearing in equation (29) does not require the evaluation of $\mu_X[k]$, $k = 0, 1, \dots, n$.

Remark 3.1 Note that $f_{Y_0}[y]$ is a shifted gamma density function with parameters $(\alpha \equiv v+1 = (\mu_Y[1]-a)^2)/(\mu_Y[2]-\mu_Y[1]^2)$ and $(\beta \equiv c = \mu_Y[2]-\mu_Y[1]^2)/(\mu_Y[1]-a)$ whose mean, $\alpha\beta+a = \mu_Y[1]$, and variance, $\alpha\beta^2 = \mu_Y[2]-\mu_Y[1]^2$, match those of Y and that, in light of equation (27), we can express $f_{Y_n}[y]$ as the product of an initial shifted gamma density approximation specified by $f_{Y_0}[y]$ times a polynomial adjustment. That is,

$$f_{Y_n}[y] = \frac{(y-a)^{\alpha-1} e^{-(y-a)/\beta}}{\beta^\alpha \Gamma[\alpha]} \sum_{j=0}^n \omega_j L_j\left(\alpha-1, \frac{y-a}{\beta}\right), \quad (30)$$

where $\omega_j = \Gamma[\alpha] \delta_j$.

□ Example 3: The Case of the Standard Lognormal Distribution

As pointed out at the beginning of this section, the proposed methodology is contraindicated when a distribution is not uniquely defined by its moments or when its tail behaviour is not that of a gamma random variable. A case in point is the lognormal distribution. As shown in Figure 5, if we employ the methodology outlined in this section, a very crude approximation of the CDF of the standard lognormal distribution is obtained on the basis of its first three moments. When additional moments are being used, the resulting density approximants turn out to be unusable.

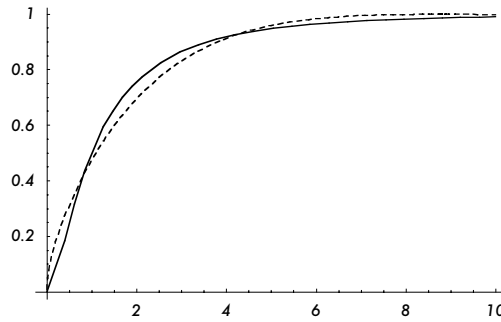


Figure 5. Exact and approximate (dashed line) CDFs. [CDFLN in the Appendix]

The following example is relevant as nonnegative definite quadratic forms in normal variables—which happen to be ubiquitous in statistics—can be expressed as mixtures of chi-square random variables, [18, Chapters 2, 7].

□ **Example 4: A Mixture of Gamma Random Variables**

Let the random variable Y be a mixture of three equally weighted, shifted gamma random variables defined on the interval $(5, \infty)$ with parameters $(\alpha_1 = 8, \beta_1)$, $(\alpha_2 = 16, \beta_2 = 1)$, and $(\alpha_3 = 64, \beta_3 = 1/2)$. The density and moment-generating functions of Y are given in the Appendix by $f_Y[y]$ and $\text{MGF}_Y[t]$, respectively. The b th moment of this distribution, denoted by $\mu_Y[b]$, is determined by evaluating the b th derivative of $\text{MGF}_Y[t]$ with respect to t at $t = 0$.

Figure 6 shows the exact density function of the mixture as well as the initial gamma density approximation given by $f_{Y_0}[y]$. Clearly, traditional approximants which make use of three or four moments could not capture adequately all the distinctive features of this particular distribution.

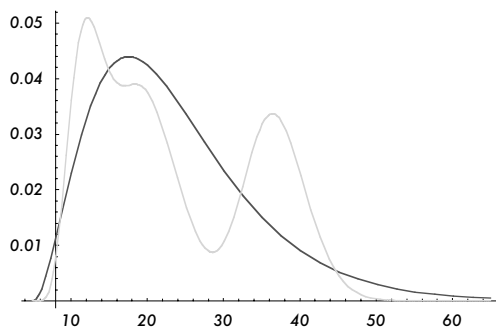


Figure 6. Exact density function and initial gamma approximant. [PGE in the Appendix]

The exact density function, $f_Y[y]$, and its approximant, $f_{Y_{60}}[y]$, evaluated from equation (29), are plotted in Figure 7. As pointed out in Remark 3.1, this density approximant results from a polynomial adjustment applied to the shifted gamma density specified by $f_{Y_0}[y]$. (Once such an approximant is obtained, a spline could be fitted in order to reduce the degree of precision that would be required in subsequent calculations.)

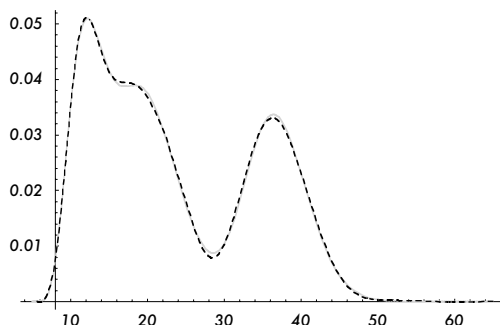


Figure 7. Exact and approximate (dashed line) PDFs. [PDEA in the Appendix]

This example illustrates that the proposed approximation formulae can also accommodate multimodal distributions and that calculations involving high-order Laguerre polynomials will readily produce remarkably accurate approximations when performed in an advanced computing environment such as that provided by *Mathematica*.

■ 4. A Unified Methodology

Remark 3.1 suggests that the exact density function associated with a distribution whose first n moments are known can be approximated by means of the product of a base density function, whose parameters are determined by matching moments, and a polynomial of degree n , whose coefficients are obtained by making use of the method of moments as well. This general semiparametric approach to density approximation, which incidentally does not rely on orthogonal polynomials, is formally described in the following result.

Result 4.1 Let $f_Y[y]$ be the density function of a continuous random variable Y defined in the interval (a, b) , $E(Y^j) \equiv \mu_Y[j]$, $X = (Y - u)/s$, where $u \in \mathbb{R}$ and $s \in \mathbb{R}^+$, $a_0 = (a - u)/s$, $b_0 = (b - u)/s$, $f_X[x] = s f_Y[u + sx]$ denote the density function of X whose support is the interval (a_0, b_0) , $E(X^j) = E[(Y - u)/s]^j \equiv \mu_X[j]$, and let the base density function $\psi_X[x] \equiv c_T w[x]$, where c_T is a positive normalizing constant, be an initial density approximant to $f_X[x]$ with $\int_{a_0}^{b_0} x^j \psi_X[x] dx \equiv m_X[j]$. Assuming that $\mu_X[i]$, $i = 0, 1, 2, \dots$, uniquely define the distribution of X , that $m_X[j]$ exists for

$j = 0, 1, \dots, 2n$, and that whenever $\psi_X[x]$ is a nontrivial function of x , its tail behaviour is congruent to that of $f_X[x]$, the latter can be approximated by

$$f_{X_n}[x] = \psi_X[x] \sum_{\ell=0}^n \xi_\ell x^\ell \quad (31)$$

with $(\xi_0, \dots, \xi_n)' = M^{-1}(\mu_X[0], \dots, \mu_X[n])'$, where M is an $(n+1) \times (n+1)$ matrix whose $(b+1)$ th row is $(m_X[b], \dots, m_X[b+n])$, $b = 0, 1, \dots, n$. When $\psi_X[x]$ depends on r parameters, these are determined by equating $m_X[j]$ to $\mu_X[j]$, $j = 1, \dots, r$. The corresponding density approximant for Y is then

$$f_{Y_n}[y] = \psi_X\left[\frac{y-u}{s}\right] \sum_{\ell=0}^n \frac{\xi_\ell}{s} \left(\frac{y-u}{s}\right)^\ell. \quad (32)$$

This last formula can easily be coded as follows:

$$\begin{aligned} f_{Y_n}[y_] &:= \psi_X\left[\frac{y_- - u}{s}\right] \\ &(\text{Inverse}[\text{Table}[m_X[b + i], \{b, 0, n\}, \{i, 0, n\}], \text{Table}[\mu_X[j], \{j, 0, n\}]] \\ &\quad \frac{\text{Table}[(\frac{y_- - u}{s})^j, \{j, 0, n\}]}{s}). \end{aligned}$$

We now show that the polynomial coefficients ξ_ℓ can be determined by making use of the method of moments, that is, by equating the first n moments obtained from $f_{X_n}[x]$ to those of X :

$$\int_{a_0}^{b_0} x^b \psi_X[x] \sum_{i=0}^n \xi_i x^i dx = \int_{a_0}^{b_0} x^b f_X[x] dx, \quad b = 0, 1, \dots, n, \quad (33)$$

which is equivalent to $(m_X[b], \dots, m_X[b+n]) \cdot (\xi_0, \dots, \xi_n) = \mu_X[b]$, $b = 0, 1, \dots, n$, where

$$\mu_X[b] = \frac{1}{s^b} \sum_{k=0}^b \binom{b}{k} \mu_Y[k] (-u)^{b-k} = \text{Expand}\left[\left(\frac{Y-u}{s}\right)^b\right] /. Y^k \rightarrow \mu_Y[k], \quad (34)$$

or $M(\xi_0, \dots, \xi_n)' = (\mu_X[0], \dots, \mu_X[n])'$, that is, $(\xi_0, \dots, \xi_n)' = M^{-1}(\mu_X[0], \dots, \mu_X[n])'$, where M is as defined in Result 4.1.

Remark 4.1 Note that it is not always necessary to transform the random variable Y . The transformation is, for instance, convenient for establishing that the proposed methodology yields density approximants identical to those obtained in terms of certain orthogonal polynomials. If there exists a base density function, $\psi_Y[y]$, whose support is the interval (a, b) and whose tail behaviour is congruent to that of Y when $\psi_Y[y]$ is a nontrivial function of y , then its parameters can be determined by equating $m_Y[j]$ to $\mu_Y[j]$ for $j = 1, \dots, r$, and $f_{Y_n}[y] = \psi_Y[y] \sum_{\ell=0}^n \xi_\ell y^\ell$ with $(\xi_0, \dots, \xi_n)' = M^{-1}(\mu_Y[0], \dots, \mu_Y[n])'$, where M is an $(n+1) \times (n+1)$ matrix whose $(b+1)$ th row is $(m_Y[b], \dots, m_Y[b+n])$, $b = 0, 1, \dots, n$. Alternatively, in that case, one may set $u = 0$ and $s = 1$ in Result 4.1.

□ Connection to Approximants Expressed in Terms of Orthogonal Polynomials

We now show that the unified methodology described in Result 4.1 provides approximants that are mathematically equivalent to those obtained from orthogonal polynomials whose associated weight function is proportional to a base density function. In addition, an alternative representation of the ξ_i 's is given in terms of c_T , the moments of X , and quantities characterizing the type of orthogonal polynomials corresponding to the selected base density function. A general representation of the coefficients η_i in the linear combination of orthogonal polynomials T_i specified by equation (36) is also derived.

Let $\{T_i[x] = \sum_{k=0}^i \delta_{ik} x^k, i = 0, 1, \dots, n\}$ be a set of orthogonal polynomials defined on the interval (a_0, b_0) , which satisfy the following orthogonality property:

$$\int_{a_0}^{b_0} w[x] T_i[x] T_b[x] dx = \theta_b \text{ when } i = b, \quad (35)$$

$$b = 0, 1, \dots, n, \text{ and zero otherwise,}$$

where $w[x]$ is a weight function, and let c_T be a normalizing constant such that $c_T w[x] \equiv \psi_X[x]$ (the base density function defined in Result 4.1) integrates to one over the interval (a_0, b_0) . On noting that the orthogonal polynomials T_i are linearly independent [14, Corollary 8.7], we can write equation (31) as

$$f_{X_n}[x] = c_T w[x] \sum_{i=0}^n \eta_i T_i[x], \quad (36)$$

where, in light of equation (33) and the fact that orthogonal polynomials are linear combinations of powers of x , the η_i 's can be obtained from equating $\int_{a_0}^{b_0} T_b[x] f_{X_n}[x] dx$ to $\int_{a_0}^{b_0} T_b[x] f[x] dx$ for $b = 0, 1, \dots, n$. This yields the following equalities:

$$c_T \int_{a_0}^{b_0} T_b[x] w[x] \sum_{i=0}^n \eta_i T_i[x] dx = \int_{a_0}^{b_0} T_b[x] f_X[x] dx, \quad b = 0, 1, \dots, n. \quad (37)$$

Equivalently, we obtain

$$\sum_{i=0}^n \eta_i c_T \int_{a_0}^{b_0} w[x] T_i[x] T_b[x] dx = \sum_{k=0}^b \delta_{hk} \mu_X[k], \quad b = 0, 1, \dots, n, \quad (38)$$

where δ_{hk} is the coefficient of x^k in $T_b[x]$ or $\delta_{hk} = \text{CoefficientList}[T_b[x], x][[k+1]]$. Thus, by virtue of the orthogonality property of the $T_i[\cdot]$'s specified by equation (35), we obtain the following general representation for the coefficients η in equation (36):

$$\eta_b = \frac{1}{c_T \theta_b} \sum_{k=0}^b \delta_{hk} \mu_X[k], \quad b = 0, 1, \dots, n, \quad (39)$$

and

$$f_{X_n}[x] = \psi[x] \sum_{i=0}^n \left(\frac{1}{c_T \theta_i} \sum_{k=0}^i \delta_{ik} \mu_X[k] \right) T_i[x]. \quad (40)$$

Now, since $T_i[x] = \sum_{\ell=0}^i \delta_{i\ell} x^\ell$ and $\sum_{i=0}^n \sum_{\ell=0}^i \equiv \sum_{\ell=0}^n \sum_{i=\ell}^n$, it follows that the coefficients ξ_ℓ in equation (31) correspond to the expression in parentheses in the following representation of $f_{X_n}[x]$:

$$f_{X_n}[x] = \psi[x] \sum_{\ell=0}^n \left(\sum_{i=\ell}^n \frac{\delta_{i\ell}}{c_T \theta_i} \sum_{k=0}^i \delta_{ik} \mu_X[k] \right) x^\ell. \quad (41)$$

Now, letting $Y = u + sX$, $Y_n = u + sX_n$, $a = u + sa_0$, $b = u + sb_0$, and denoting by $f_Y[y]$ and $f_{Y_n}[y]$ the density functions of Y and Y_n corresponding to those of X and X_n , respectively, we can approximate $f_Y[y]$ whose support is the interval (a, b) by

$$f_{Y_n}[y] = w\left[\frac{y-u}{s}\right] \sum_{i=0}^n \left(\frac{1}{s \theta_i} \sum_{k=0}^i \delta_{ik} \mu_X[k] \right) T_i\left[\frac{y-u}{s}\right], \quad (42)$$

where $\sum_{k=0}^i \delta_{ik} \mu_X[k] \equiv T_i[X] / X^i \rightarrow \mu_X[j] \equiv T_i[(Y-u)/s] / Y^i \rightarrow \mu_Y[j]$ or equivalently

$$f_{Y_n}[y] := \psi\left[\frac{y-u}{s}\right] \sum_{\ell=0}^n \left(\sum_{i=\ell}^n \frac{\delta_{i\ell}}{s c_T \theta_i} \sum_{k=0}^i \delta_{ik} \mu_X[k] \right) \left(\frac{y-u}{s}\right)^\ell, \quad (43)$$

which corresponds to the representation of $f_{Y_n}[y]$ given in equation (32).

It should be pointed out that Reinking proposed under a somewhat different setup general formulae for approximating density and distribution functions in terms of Laguerre, Jacobi, and Hermite polynomials [19]. Arguably, the approach proposed in Result 4.1 is not only conceptually simpler than that which is based on orthogonal polynomials, but it is also more general. The particular cases of density approximants expressed in terms of Laguerre, Legendre, Jacobi, and Hermite polynomials, which can all be equivalently obtained via Result 4.1, are individually considered in the remainder of this section.

Approximants Based on Laguerre Polynomials

Consider the approximants based on the Laguerre polynomials which were defined in Section 3. In that case, $u = a$, $s = c$, $Y = cX + a$, $a_0 = 0$, $b_0 = b = \infty$, $w[x] = x^v e^{-x}$, $c_T = 1/\Gamma[v+1]$, $T_i[x]$ is a $\text{LaguerreL}[i, v, x]$ orthogonal polynomial which is defined on the interval $[0, \infty)$, and θ_b , as specified by equation (35), is equal to $\Gamma[v+b+1]/b! = \text{Pochhammer}[b+1, v]$. It is then easily seen that the density expressions given in equations (42) and (28) coincide.

In this case, the base density function $\psi_X[x]$ is that of a gamma random variable with parameters $v+1$ and 1. Note that after applying the transformation

$Y = cX + a$, the base density becomes a shifted gamma distribution with parameters $v + 1$ and c whose support is the interval $[a, \infty)$.

Alternatively, we can obtain an identical density approximant by making use of Result 4.1, wherein $\psi_X[x]$ is a gamma density function with parameters $v + 1$ and 1, whose j th moment $m_X[j]$ which is needed to determine the ξ_ℓ 's, is given by $\text{Gamma}[v + 1 + j] / \text{Gamma}[v + 1] = \text{Pochhammer}[v + 1, j]$, $j = 0, 1, \dots, n$.

For instance, Laguerre series expansions were respectively derived by [20] and [21] for approximating the density functions of quadratic forms in normal variables and those of noncentral χ^2 and F random variables.

Approximants Based on Legendre Polynomials

First, we note that whenever the finite interval $[a, b]$ is mapped onto the interval $[a_0, b_0]$, the requisite affine transformation is

$$X = \frac{Y - u}{s} \quad (44)$$

with $u = (a b_0 - a_0 b) / (b_0 - a_0)$ and $s = (b - a) / (b_0 - a_0)$.

Now, considering the approximants based on Legendre polynomials discussed in Section 2, which are defined on the interval $[-1, 1]$, we have $u = (a + b) / 2$, $s = (b - a) / 2$, $w[x] = 1$, $c_T = 1 / 2$, $T_i[x]$ is a LegendreP $[i, x]$ orthogonal polynomial, and $\theta_b = 2 / (2b + 1)$. It is then easily seen that $f_{Y_n}[y]$ as given in equation (42) yields the density function of Y_n specified in equation (13). The same density approximant can also be obtained by making use of Result 4.1. In this case, $m_X[j] = (1 - (-1)^{j+1}) / (2(j + 1))$, $j = 0, 1, \dots$.

Approximants Based on Jacobi Polynomials

In order to approximate densities for which a beta density function is suitable as a base density function, we shall make use of the following alternative form of the Jacobi polynomials:

$$G_{n-}[\sigma_-, \tau_-, x_-] := n! \frac{\text{Gamma}[n + \sigma]}{\text{Gamma}[2n + \sigma]} \text{JacobiP}[n, \sigma - \tau, \tau - 1, 2x - 1] \quad (45)$$

defined on the interval $[0, 1]$, wherein $\text{JacobiP}[n, a_1, b_1, w]$ denotes an n th-degree Jacobi polynomial in the variable w with parameters a_1 and b_1 , $\sigma = \alpha + \beta + 1$, and $\tau = \alpha + 1$. In this case, the support of Y , the random variable of interest, is the finite interval $[a, b]$; thus $u = a$ and $s = b - a$ in the linear transformation leading to equation (42). The associated weight function is $w[x] = x^\alpha (1 - x)^\beta$ so that $c_T = 1 / \text{Beta}[\alpha + 1, \beta + 1]$, and the base density is that of a $\text{Beta}(\alpha + 1, \beta + 1)$ random variable, that is,

$$\psi_X[x] := \frac{1}{\text{Beta}[\alpha + 1, \beta + 1]} x^\alpha (1 - x)^\beta, \quad 0 < x < 1, \quad (46)$$

whose j th moment is given by

$$m_X[j] = \frac{\text{Gamma}[\alpha + \beta + 2] \text{Gamma}[\alpha + 1 + j]}{\text{Gamma}[\alpha + 1] \text{Gamma}[\alpha + j + \beta + 2]} = \frac{\text{Pochhammer}[\alpha + 1, j]}{\text{Pochhammer}[\alpha + \beta + 2, j]}, \quad j = 0, 1, \dots \quad (47)$$

The parameters α and β can be determined as follows:

$$\begin{aligned} \alpha &= \mu_X[1] \frac{\mu_X[1] - \mu_X[2]}{\mu_X[2] - \mu_X[1]^2} - 1, \\ \beta &= (1 - \mu_X[1]) \frac{\alpha + 1}{\mu_X[1]} - 1, \end{aligned} \quad (48)$$

see [22, 44] and, in this case,

$$\theta_k^{-1} = \frac{(2k + \alpha + \beta + 1) \text{Gamma}[2k + \alpha + \beta + 1]^2}{k! \text{Gamma}[k + \alpha + 1] \text{Gamma}[k + \alpha + \beta + 1] \text{Gamma}[k + \beta + 1]}, \quad (49)$$

[19]. As illustrated in the next example, Result 4.1, as well as equations (42) and (43), yields identical density approximants.

Jacobi polynomial expansions were used by Durbin and Watson to approximate certain percentiles of their well-known test statistic [23].

□ Example 5

Example 1 is revisited by making use of a beta density as a base density function in Result 4.1 or equivalently by resorting to an approximant expressed in terms of Jacobi polynomials. Since the base density function already provides a good approximation in this case, fewer moments are required than in the case of the Legendre polynomial approximant (eight as opposed to 13) in order to achieve a similar degree of accuracy.

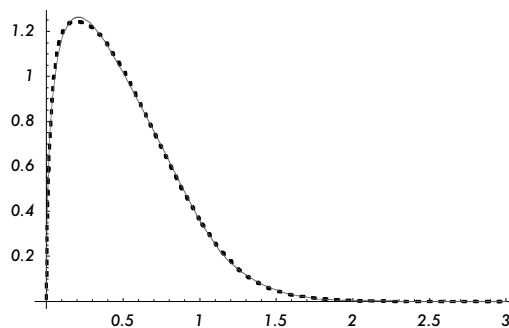


Figure 8. Exact and approximate (dashed line) PDFs. [S5 in the Appendix]

□ Example 6

Consider Wilks' likelihood ratio statistic, $\Lambda = |S_e|^{N/2} / |S_e + S_b|^{N/2}$, where S_e is the error sum of squares matrix and S_b is the hypothesis sum of squares matrix, which is used for testing linear hypotheses on regression coefficients on the basis of N p -dimensional observation vectors. Assuming standard normal theory, S_e and S_b are independent Wishart matrices. As shown in [24, Section 8.4], when the mean vectors are assumed to be of the form BZ_α , where $B = (B_1, B_2)$ with B_i of dimension $p \times q_i$, $i = 1, 2$, and the Z_α 's are given $(q_1 + q_2)$ -dimensional vectors, $\alpha = 1, \dots, N$, the b th moment of the statistic $U = \Lambda^{2/N}$ for testing the null hypothesis that B_1 is a given matrix is

$$\prod_{i=1}^p \frac{\text{Gamma}\left[\frac{N-q_1-q_2+1-i}{2} + b\right] \text{Gamma}\left[\frac{N-q_2+1-i}{2}\right]}{\text{Gamma}\left[\frac{N-q_1-q_2+1-i}{2}\right] \text{Gamma}\left[\frac{N-q_2+1-i}{2} + b\right]}. \quad (50)$$

Clearly, the support of Λ , and thus that of U , is the interval $(0, 1)$. As pointed out by Mathai [25] who obtained a general representation of the exact density function of U as an inverse Mellin transform, a wide array of techniques were used to determine the density of U for some particular values of N , q_1 , q_2 and p . Result 4.1 provides a means of obtaining approximants that can be viewed as exact for all intents and purposes. In any case, many of the so-called exact representations available in the literature are expressed in terms of integrals which have to be evaluated numerically or in terms of infinite series that have to be truncated.

Figures 9 and 10 show the approximate density and distribution functions of U for $N = 12$, $q_1 = 2$, $q_2 = 4$, and $p = 2$ when one lets $n = 2$ (dashed line) and $n = 20$ (solid line). Mathai [25] determined that for $N - q_1 - q_2 = 6$, $q_1 = 2$, and $p = 2$, the 95th and 99th percentiles of the distribution are respectively 0.87825 and 0.94719. As evaluated in the Appendix, $F1_{Y_{20}}[0.87825] = 0.950001$ and $F1_{Y_{20}}[0.94719] = 0.990001$, where $F1_{Y_n}[w]$ denotes a percentile corresponding to the point w , which is approximated on the basis of n moments. It is seen from the graph that an adequate approximant can be obtained from two moments in this case. In fact, $F1_{Y_2}[0.87825] = 0.9507$ and $F1_{Y_2}[0.94719] = 0.9903$.

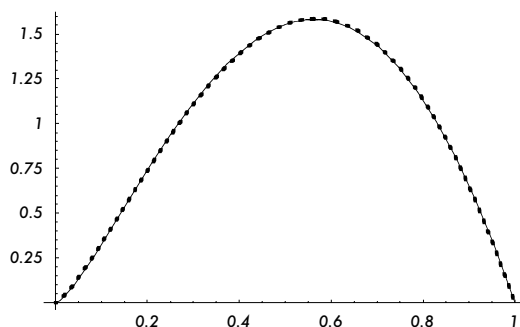


Figure 9. PDF of U for $n = 2$ (dashed line) and $n = 20$. [P6 in the Appendix]

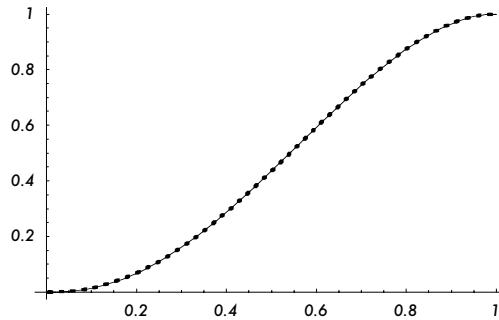


Figure 10. CDF of U for $n = 2$ (dashed line) and $n = 20$. [C6 in the Appendix]

Approximants Based on Hermite Polynomials

We can approximate densities whose tail behaviour is congruent to that of a normal density function by means of the modified Hermite polynomials given by

$$H_{k-}^*[x_-] := (-1)^k 2^{-k/2} \text{HermiteH}\left[k, \frac{x}{\sqrt{2}}\right] \quad (51)$$

and defined on the interval $(-\infty, \infty)$, where $\text{HermiteH}[k, w]$ denotes a k th-degree Hermite polynomial in the variable w in *Mathematica* notation. The weight function associated with modified Hermite polynomials which is $w[x] = e^{-x^2/2}$ is proportional to the density function of a standard Gaussian random variable. Thus, the requisite transformation is $X = (Y - u)/s$ with $u = \mu_Y[1]$ and $s = \sqrt{\mu_Y[2] - \mu_Y[1]^2}$, the normalizing constant is $c_T = 1/\sqrt{2\pi}$, and the base density function is that of a standard normal random variable, that is,

$$\psi_X[x] = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}, \quad -\infty < x < \infty, \quad (52)$$

whose j th moment is given by

$$m_X[j] = \frac{2^{1/2} (-1+j) (1 + (-1)^j) \text{Gamma}\left[\frac{1+j}{2}\right]}{\sqrt{2\pi}}, \quad j = 0, 1, \dots \quad (53)$$

Moreover, in this case,

$$\theta_k = \sqrt{2\pi} k!. \quad (54)$$

The same density approximants are obtained whether one makes use of equation (32) or (42). They are also known as (type-A) Gram-Charlier expansions. A methodology for determining the Hermite polynomial coefficients in the expansions in terms of the moments is presented in [3, Section 5.4], where the advantages and drawbacks of using such approximations are also discussed. Conditions ensuring the convergence of the approximants are available from Cramér [26].

□ Example 7

Consider an equally weighted mixture of two Gaussian distributions with parameters (3, 4) and (1, 1), whose density function is plotted as a solid line in Figure 11. On the basis of 30 moments, identical density approximants (represented by a dashed line) are obtained whether we make use of equation (42) or Result 4.1.

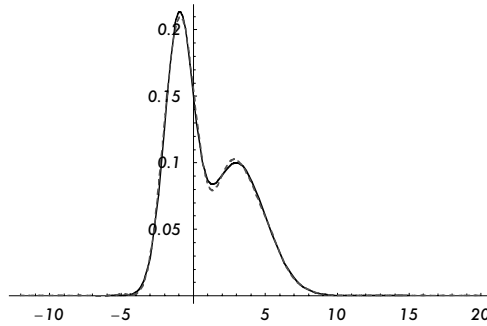


Figure 11. Exact and approximate (dashed line) PDFs. [S7, S71, or S7a in the Appendix]

■ Acknowledgments

I am very grateful to a reviewer for several helpful suggestions which improved the presentation of the material. Thanks are also due to the guest editor and a second reviewer for pointing out the work of Reinking [19]. Finally, the financial support of the Natural Sciences and Engineering Research Council of Canada is gratefully acknowledged.

■ Appendix

This appendix contains the *Mathematica* code that was used in connection with each of the seven examples. All the calculations are carried out with rational numbers so as to prevent any loss of precision. *It would be advisable to quit the kernel between examples.*

□ Code for Example 1

The end points of the support of the random variable are a and b . The degree of the polynomial approximation is n . In this example, the random variable of interest is V , which will be denoted by Y to conform to the general notation utilised in Section 4. Its b th moment is given by $\mu_Y[b]$ whereas the j th moment of X defined on the interval $[-1, 1]$ is denoted by $\mu_X[j]$. The exact density function is denoted by $f_Y[y]$ and the approximate density functions obtained

from equations (13) and (15) are given by $f_{Y_n}[y]$ and $f_{1Y_n}[y]$, respectively. The exact and approximate cumulative distribution functions obtained by integration are denoted by $Q_Y[u]$ and $Q_{Y_n}[u]$, respectively.

```

a = 0;
b = 3;
n = 13;
μY[h_] := μY[h] =
  ∫01 ∫01 ∫01 (δ1 + δ2 + δ3)h (δ1-1/2 - 1) (δ2-1/2 - 1) (δ3-1/2 - 1) dδ1 dδ2 dδ3;

μX[j_] :=
  μX[j] = Expand[ ( (2 Y - (a + b)) / (b - a) )j ] /. Table[Yi → μY[i], {i, 0, n}];

fY[y_] := fY[y] = Which[0 ≤ y < 1, π (2 √y - 3 y) + 4 y3/2 - y2 / 2,
  1 ≤ y ≤ 2, -1/2 + π (3 - 4 √y + 6 y) + 3 y + y2 -
  4 √-1 + y (1 + 2 y) - 12 y ArcSin[1/√y], 2 < y ≤ 3,
  4 √-2 + y (1 + y) + 1/2 (-5 + π (2 - 6 y) - y (6 + y)) +
  4 (-1 + 3 y) ArcCsc[√-1 + y] + 8 ArcTan[ (3 - y) / (2 √-2 + y) ] -
  4 √y ( ArcTan[1 / (√(-2 + y) y)] + ArcTan[ (1 - (-2 + y) y) / (2 √(-2 + y) y) ] ) ];

fYn[y_] := ∑k=0n ( (2 k + 1) / (b - a) LegendreP[k, X] /. Xj → μX[j] )
  LegendreP[k, (2 y - (a + b)) / (b - a)];

f1Yn[y_] := ∑k=0n ( (2 k + 1) / (b - a) ( LegendreP[k, (2 Y - (a + b)) / (b - a)] /. Yj → μY[j] )
  LegendreP[k, (2 y - (a + b)) / (b - a)];

Pq = Plot[{fYn[y], fY[y]}, {y, 0, 3},
  PlotStyle → {{Thickness[0.00765], Dashing[{0.005, 0.019]}},
    {Thickness[0.001], Hue[0.7]}}];

Pq1 = Plot[{f1Yn[y], fY[y]}, {y, 0, 3},
  PlotStyle → {{Thickness[0.00765], Dashing[{0.005, 0.019]}},
    {Thickness[0.001], Hue[0.7]}}];

```

```

QY[u_] := ∫0u fY[y] dy;

QYn[u_] := QYn[u] = ∫0u fYn[y] dy;

PQ = Plot[{QYn[u], QY[u]}, {u, 0, 3}, PlotRange → All,
  PlotStyle → {{Thickness[0.00765'], Dashing[{0.025', 0.029'}]},
    {Thickness[0.001', Hue[0.7']}}];

Qd = Plot[QYn[u] - QY[u], {u, 0, 3}, PlotRange → All];

```

□ Code for Example 2

The end points of the support of the random variable are a and b . The degree of the polynomial approximation is n . For a given sequence of moments, $\mu_Y[b]$, $b = 1, 2, \dots, n$, the approximate density, $f_{Y_n}[y]$, is obtained from equation (14) in this case.

```

ClearAll[a, b, n, μ, f];
a = 0;
b = 1;
n = 15;

<< Statistics`ContinuousDistributions`

μY[h_] := μY[h] = ∫01 xh  $\frac{1}{2}$  (PDF[BetaDistribution[3, 2], x] +
  PDF[BetaDistribution[2, 30], x]) dx;

fYn[y_] :=
  ∑k=0n  $\frac{2k+1}{b-a}$  ⎛ LegendreP[k, X] /. Xj → ∑k=0j  $\frac{j! 2^k \mu_Y[k] (-a-b)^{j-k}}{(b-a)^j k! (j-k)!}$  ⎞
  LegendreP[k,  $\frac{2y-(a+b)}{b-a}$ ];

Pb = Plot[{ $\frac{1}{2}$  (PDF[BetaDistribution[3, 2], y] +
  PDF[BetaDistribution[2, 30], y]), fYn[y]},
  {y, 0, 1}, PlotRange → All, PlotStyle →
  {{RGBColor[0, 1, 0]}, {Dashing[{0.01, 0.01, 0.01, 0.01}]}}];

```

□ Code for Example 3

The b th moment of the standard lognormal distribution whose support is the positive half-line is $\mu_Y[b] = e^{b^2/2}$. The exact and approximate density functions

are respectively given by $f_Y[y]$ and $f_{Y_n}[y]$, the latter being obtained from equation (28). $F_{Y_n}[w]$ denotes the distribution function corresponding to $f_{Y_n}[y]$.

```

a = 0;
n = 3;

<< Statistics`ContinuousDistributions`

f_Y[y_] := f_Y[y] = PDF[LogNormalDistribution[0, 1], y];
mu_Y[h_] := mu_Y[h] = ExpectedValue[y^h, LogNormalDistribution[0, 1], y];

c = (mu_Y[2] - mu_Y[1]^2) / (mu_Y[1] - a);

v = -1 + (mu_Y[1] - a) / c;

mu_X[j_] := mu_X[j] = Expand[(Y - a) / c]^j /. Y^h_ -> mu_Y[h];

f_Yn[y_] := f_Yn[y] = ((y - a)^v * Exp[-(y - a)/c]) / c^(v+1)
Sum[j! LaguerreL[j, v, (y - a)/c] / Gamma[v + j + 1], {j, 0, n}]
(LaguerreL[j, v, X] /. X^k_ -> mu_X[k]);

PE = Plot[f_Y[y], {y, 0, 10},
  PlotRange -> All, PlotStyle -> RGBColor[0, 0, 1]];
PA = Plot[f_Yn[y], {y, 0.1, 10}, PlotRange -> All,
  PlotStyle -> Dashing[{0.01, 0.01, 0.01, 0.01}]];
PEA = Show[PE, PA];
CLN = Plot[CDF[LogNormalDistribution[0, 1], y], {y, 0, 10}];

F_Yn[w_] := F_Yn[w] = Integrate[f_Yn[y] dy, {y, 0, w}];

CLN3 = Plot[Evaluate[F_Yn[w]], {w, 0, 10}, PlotRange -> All,
  PlotStyle -> Dashing[{0.01, 0.01, 0.01, 0.01}]];
CDFLN = Show[CLN3, CLN];

```

□ Code for Example 4

The b th moment of the mixture of three shifted gamma random variables, which is denoted by $\mu_Y[b]$, is obtained by differentiation of the moment-generating function, $\text{MGF}_Y[t]$. The degree of the polynomial approximation is n . The exact and approximate density functions are respectively given by $f_Y[y]$ and $f_{Y_n}[y]$, the latter being determined from equation (29).

```

a = 5;
fY[y_] := If[y > a,
  
$$\frac{(y-a)^{\alpha_1-1} e^{-(y-a)/\beta_1}}{3 \text{Gamma}[\alpha_1] \beta_1^{\alpha_1}} + \frac{(y-a)^{\alpha_2-1} e^{-(y-a)/\beta_2}}{3 \text{Gamma}[\alpha_2] \beta_2^{\alpha_2}} + \frac{(y-a)^{\alpha_3-1} e^{-(y-a)/\beta_3}}{3 \text{Gamma}[\alpha_3] \beta_3^{\alpha_3}}, 0];$$


α1 = 8; β1 = 1;
α2 = 16; β2 = 1;
α3 = 64; β3 = 1/2;

n = 60;
MGFY[t_] = ea t  $\frac{(1-\beta_1 t)^{-\alpha_1}}{3}$  + ea t  $\frac{(1-\beta_2 t)^{-\alpha_2}}{3}$  + ea t  $\frac{(1-\beta_3 t)^{-\alpha_3}}{3}$ ;
μY[h_] := μY[h] = D[MGFY[t], {t, h}] /. t → 0;
c =  $\frac{\mu_Y[2] - \mu_Y[1]^2}{\mu_Y[1] - a}$ ;
v = -1 +  $\frac{\mu_Y[1] - a}{c}$ ;
fYn[y_] := fYn[y] =  $\frac{(y-a)^v e^{-(y-a)/c}}{c^{v+1}}$ 

$$\sum_{j=0}^n \frac{j!}{\text{Gamma}[v+j+1]} \left( \text{LaguerreL}[j, v, \frac{y-a}{c}] / . Y^{k-} \Rightarrow \mu_Y[k] \right)$$

* LaguerreL[j, v,  $\frac{y-a}{c}$ ];

ψY[y_] := ψY[y] =  $\frac{(y-a)^v e^{-(y-a)/c}}{c^{v+1} \text{Gamma}[v+1]}$ ;

PE = Plot[fY[y], {y, a, 65},
  PlotRange → All, PlotStyle → RGBColor[0, 1, 0]];
PA = Plot[fYn[y], {y, a, 65}, PlotRange → All,
  PlotStyle → Dashing[{0.01, 0.01, 0.01, 0.01}]];
PDEA = Show[PE, PA];
PG = Plot[ψY[y], {y, a, 65},
  PlotRange → All, PlotStyle → RGBColor[0, 0, 1]];
PGE = Show[PG, PE];

```

□ Code for Example 5

We use the notation introduced in Section 4. First, we approximate the density function of the random variable V as defined in Section 2 by making use of equation (42) with $T_i[x] \equiv G_i[\alpha + \beta + 1, \alpha + 1, x]$, a modified Jacobi polynomial. To conform to the general notation used in Section 4, we will again replace V with Y .

```

u = 0;
s = 3;
n = 8;
fy[y_] :=

fy[y] = Which[0 ≤ y < 1, π (2 √y - 3 y) + 4 y3/2 -  $\frac{y^2}{2}$ , 1 ≤ y ≤ 2, - $\frac{1}{2}$  +

π (3 - 4 √y + 6 y) + 3 y + y2 - 4 √-1 + y (1 + 2 y) - 12 y ArcSin[ $\frac{1}{\sqrt{y}}$ ],

2 < y ≤ 3, 4 √-2 + y (1 + y) +  $\frac{1}{2}$  (-5 + π (2 - 6 y) - y (6 + y)) +

4 (-1 + 3 y) ArcCsc[√-1 + y] + 8 ArcTan[ $\frac{3 - y}{2 \sqrt{-2 + y}}$ ] -

4 √y (ArcTan[ $\frac{1}{\sqrt{(-2 + y) y}}$ ] + ArcTan[ $\frac{1 - (-2 + y) y}{2 \sqrt{(-2 + y) y}}$ ])];

μy[h_] := μy[h] =

∫01 ∫01 ∫01 (δ1 + δ2 + δ3)h (δ1-1/2 - 1) (δ2-1/2 - 1) (δ3-1/2 - 1) dδ1 dδ2 dδ3;

μx[j_] := μx[j] = Expand[( $\frac{y - u}{s}$ )j] /. Yk -> μy[k];

w[x_] := xα (1 - x)β;

α = μx[1]  $\frac{\mu_x[1] - \mu_x[2]}{\mu_x[2] - \mu_x[1]^2}$  - 1;

β = (1 - μx[1])  $\frac{\alpha + 1}{\mu_x[1]}$  - 1;

Gm[σ_, τ_, x_] := m!  $\frac{\text{Gamma}[m + \sigma]}{\text{Gamma}[2 m + \sigma]}$  JacobiP[m, σ - τ, τ - 1, 2 x - 1];

θk := θk =  $\frac{1}{\frac{(2 k + \alpha + \beta + 1) \text{Gamma}[2 k + \alpha + \beta + 1]^2}{k! \text{Gamma}[k + \alpha + 1] \text{Gamma}[k + \alpha + \beta + 1] \text{Gamma}[k + \beta + 1]}}$ ;

fyn[y_] := fyn[y] = w[ $\frac{y - u}{s}$ ]

∑i=0n (  $\frac{1}{s \theta_i}$  ∑k=0i (CoefficientList[Gi[α + β + 1, α + 1, X], X][[k + 1]] μx[k])

Gi[α + β + 1, α + 1,  $\frac{y - u}{s}$ ] );

S5 = Plot[{Evaluate[fyn[y]], fy[y]}, {y, 0, 3},

PlotStyle -> {{Thickness[0.00765], Dashing[{0.005, 0.019]}},

{Thickness[0.001], Hue[0.7]}}];

```

Identical density approximants denoted by $f1_{Y_n}[y]$ and $f2_{Y_n}[y]$ in the following code are obtained from Result 4.1 and equation (43), respectively.

```


$$\psi_x[x_] := \frac{1}{\text{Beta}[\alpha + 1, \beta + 1]} x^\alpha (1 - x)^\beta;$$



$$m_x[j_] := m_x[j] = \frac{\text{Pochhammer}[\alpha + 1, j]}{\text{Pochhammer}[\alpha + \beta + 2, j]};$$



$$\text{IMb}[x_] := (\text{Inverse}[\text{Table}[m_x[h + i], \{h, 0, n\}, \{i, 0, n\}]] \cdot \text{Table}[\mu_x[j], \{j, 0, n\}]) \cdot \text{Table}[x^j, \{j, 0, n\}];$$



$$f1_{Y_n}[y_] := \frac{\psi_x\left[\frac{y-u}{s}\right] \text{IMb}\left[\frac{y-u}{s}\right]}{s};$$



$$c_T = \frac{1}{\text{Beta}[\alpha + 1, \beta + 1]};$$



$$f2_{Y_n}[y_] := f2_{Y_n}[y] = \psi_x\left[\frac{y-u}{s}\right]$$


$$\sum_{l=0}^n \left( \sum_{i=l}^n \frac{1}{s c_T \theta_i} \text{CoefficientList}[G_i[\alpha + \beta + 1, \alpha + 1, X], X][[l + 1]] \right.$$


$$\left. \sum_{k=0}^i (\text{CoefficientList}[G_i[\alpha + \beta + 1, \alpha + 1, X], X][[k + 1]] \mu_x[k]) \right) \left( \frac{y-u}{s} \right)^l;$$


Simplify[fYn[y] // N]

Simplify[f1Yn[y] // N]

Simplify[f2Yn[y] // N]

```

□ Code for Example 6

We use Result 4.1 to approximate the density of Wilks' likelihood ratio criterion, which will be denoted by Y to conform to the general notation utilised in Section 4. In this case, both X and Y are defined on the interval $(0, 1)$. The transformation is included in the code so that it could be applied if needed. The symbol N being protected, it is replaced by N^* in the following code.

```

u = 0;
s = 1;
N* = 12;
q1 = 2;
q2 = 4;
p = 2;

μY[h_] := μY[h] = 
$$\prod_{i=1}^p \frac{\text{Gamma}\left[\frac{N^*-q_1-q_2+1-i}{2} + h\right] \text{Gamma}\left[\frac{N^*-q_2+1-i}{2}\right]}{\text{Gamma}\left[\frac{N^*-q_1-q_2+1-i}{2}\right] \text{Gamma}\left[\frac{N^*-q_2+1-i}{2} + h\right]};$$


μX[j_] := μX[j] = Expand $\left[\left(\frac{Y-u}{s}\right)^j\right] /. Y^{k_-.} \rightarrow \mu_Y[k];$ 

α = μX[1]  $\frac{\mu_X[1] - \mu_X[2]}{\mu_X[2] - \mu_X[1]^2} - 1;$ 

β = (1 - μX[1])  $\frac{\alpha + 1}{\mu_X[1]} - 1;$ 

ψX[x_] :=  $\frac{1}{\text{Beta}[\alpha + 1, \beta + 1]} x^\alpha (1 - x)^\beta;$ 

mX[j_] := mX[j] =  $\frac{\text{Pochhammer}[\alpha + 1, j]}{\text{Pochhammer}[\alpha + \beta + 2, j]};$ 

IMb[x_, n_] :=
  IMb[x, n] = (Inverse[Table[mX[h + i], {h, 0, n}, {i, 0, n}]] .
    Table[μX[j], {j, 0, n}]) . Table[xj, {j, 0, n}];

f1Yn[y_] :=  $\frac{\psi_X\left[\frac{y-u}{s}\right] \text{IMb}\left[\frac{y-u}{s}, n\right]}{s};$ 

P6 = Plot[{Evaluate[f1Y2[y]], Evaluate[f1Y20[y]]}, {y, u, u + s},
  PlotStyle → {{Thickness[0.00765], Dashing[{0.005, 0.019]}},
    {Thickness[0.001]}}];

F1Yn[w_] := F1Yn[w] =  $\int_u^w f1_{Y_n}[y] dy;$ 

C6 = Plot[{Evaluate[F1Y2[w]], Evaluate[F1Y20[w]]}, {w, u, u + s},
  PlotStyle → {{Thickness[0.00765], Dashing[{0.005, 0.019]}},
    {Thickness[0.001]}}];

F1Y20[0.87825] // N

F1Y20[0.94719] // N

F1Y2[0.87825] // N

F1Y2[0.94719] // N

```

□ Code for Example 7

We use the notation introduced in Section 4. First, we approximate the density function of the Gaussian mixture with $f_{Y_n}[y]$ by making use of an alternate representation of equation (42) with $T_i[x] \equiv H^*_i[x]$, a modified Hermite polynomial. The function $f1_{Y_n}[y]$ provides an approximant which is directly expressed in terms of the moments of Y .

```

fY[y_] := fY[y] =  $\frac{1}{2} \frac{1}{\sqrt{2\pi}} \frac{1}{2} (e^{-(y-3)^2/8}) + \frac{1}{2} \frac{1}{\sqrt{2\pi}} (e^{-(y+1)^2/2});$ 

μY[h_] :=
 $\frac{1}{2} \int_{-\infty}^{\infty} y^h \frac{1}{\sqrt{2\pi}} \frac{1}{2} (e^{-(y-3)^2/8}) dy + \frac{1}{2} \int_{-\infty}^{\infty} y^h \frac{1}{\sqrt{2\pi}} (e^{-(y+1)^2/2}) dy;$ 

n = 30;
Table[μY[h], {h, 0, n}];
u = μY[1];
s =  $\sqrt{\mu_Y[2] - \mu_Y[1]^2};$ 

μX[j_] := μX[j] = Expand[ $(\frac{Y-u}{s})^j$ ] /. Yh -> μY[h];

w[x_] := e-x2/2;

H*k[x_] := (-1)k 2-k/2 HermiteH[k,  $\frac{x}{\sqrt{2}}$ ];

θk := θk =  $\sqrt{2\pi} k!$ ;

fYn[y_] := w[ $\frac{y-u}{s}$ ]  $\sum_{i=0}^n \left( \frac{1}{s \theta_i} (H^*_i[x] /. x^{j-} -> \mu_X[j]) H^*_i[\frac{y-u}{s}] \right);$ 

f1Yn[y_] := w[ $\frac{y-u}{s}$ ]  $\sum_{i=0}^n \left( \frac{1}{s \theta_i} (H^*_i[x] /. x^{j-} -> \text{Expand}[(\frac{Y-u}{s})^j] /. Y^h -> \mu_Y[h]) H^*_i[\frac{y-u}{s}] \right);$ 

S7 = Show[Plot[fY[y], {y, -6, 15}, PlotRange -> All],
Plot[Evaluate[fYn[x]], {x, -12, 20}, PlotRange -> All,
PlotStyle -> {Dashing[{0.01, 0.01}], RGBColor[0, 0, 1]}]];
S71 = Show[Plot[fY[y], {y, -6, 15}, PlotRange -> All],
Plot[Evaluate[f1Yn[x]], {x, -12, 20}, PlotRange -> All,
PlotStyle -> {Dashing[{0.01, 0.01}], RGBColor[0, 0, 1]}]];

```

The same density approximant is obtained from equation (32).

```

n = 30;

$$\psi_x[x_] := \frac{1}{\sqrt{2\pi}} e^{-x^2/2};$$


$$m_x[j_] := m_x[j] = \frac{2^{1/2(-1+j)} (1 + (-1)^j) \text{Gamma}\left[\frac{1+j}{2}\right]}{\sqrt{2\pi}};$$

IMb[x_] := (Inverse[Table[m_x[h+i], {h, 0, n}, {i, 0, n}]] .
  Table[m_x[j], {j, 0, n}]).Table[x^j, {j, 0, n}];

$$f2_{v_n}[y_] := \frac{\psi_x\left[\frac{y-u}{s}\right] \text{IMb}\left[\frac{y-u}{s}\right]}{s};$$

S7a = Show[Plot[f_y[y], {y, -6, 15}, PlotRange -> All],
  Plot[Evaluate[f2_{v_n}[x]], {x, -12, 20}, PlotRange -> All,
    PlotStyle -> {Dashing[{0.01, 0.01}], RGBColor[0, 0, 1]}];

```

■ References

- [1] H. Solomon and M. Stephens, "Approximations to Density Functions Using Pearson Curves," *Journal of the American Statistical Association*, **73**(361), 1978 pp. 153–160.
- [2] W. P. Elderton and N. L. Johnson, *Systems of Frequency Curves*, Oxford: Cambridge University Press, 1969.
- [3] C. Rose and M. D. Smith, *Mathematical Statistics with Mathematica*, New York: Springer-Verlag, 2002.
- [4] N. Reid, "Saddlepoint Methods and Statistical Inference," *Statistical Science*, **3**(2), 1988 pp. 213–238.
- [5] A. M. Mathai and R. K. Saxena, *The H-function with Applications in Statistics and Other Disciplines*, New York: John Wiley & Sons, 1978.
- [6] S. B. Provost and E. M. Rudiuk, "Moments and Densities of Test Statistics for Covariance Structures," *International Journal of Mathematical and Statistical Sciences*, **4**(1), 1995 pp. 85–104.
- [7] P. J. Smith, "A Recursive Formulation of the Old Problem of Obtaining Moments from Cumulants and Vice Versa," *The American Statistician*, **49**, 1995 pp. 217–219.
- [8] S. B. Provost and Y.-H. Cheong, "On the Distribution of Linear Combinations of the Components of a Dirichlet Random Vector," *The Canadian Journal of Statistics*, **28**, 2000 pp. 417–425.
- [9] G. Sansone, *Orthogonal Functions*, Pure and Applied Mathematics, Vol. 9, New York: Interscience Publishers, 1959.
- [10] G. Alexits, *Convergence Problems of Orthogonal Series*, New York: Pergamon Press, 1961.
- [11] L. Devroye and L. Györfi, *Nonparametric Density Estimation: The L_1 View*, New York: John Wiley & Sons, 1985.
- [12] W. B. Jones and A. S. Ranga, eds., *Orthogonal Functions, Moment Theory, and Continued Fractions, Theory and Applications*, Lecture Notes in Pure and Applied Mathematics, New York: Marcel Dekker, 1998.

- [13] L. Devroye, "On Random Variate Generation When Only Moments or Fourier Coefficients Are Known," *Mathematics and Computers in Simulation*, **31**, 1989 pp. 71–89.
- [14] R. L. Burden and J. D. Faires, *Numerical Analysis*, 4th ed., Boston: PWS-Kent, 1989.
- [15] C. R. Rao, *Linear Statistical Inference and Its Applications*, New York: John Wiley & Sons, 1965.
- [16] A. M. Mathai, *An Introduction to Geometric Probability: Distributional Aspects with Applications*, The Netherlands: Gordon and Breach, 1999.
- [17] G. Szegő, *Orthogonal Polynomials*, Providence, RI: American Mathematical Society, 1959.
- [18] A. M. Mathai and S. B. Provost, *Quadratic Forms in Random Variables, Theory and Applications*, New York: Marcel Dekker, 1992.
- [19] J. T. Reinking, "Orthogonal Series Representations of Probability Density and Distribution Functions," *The Mathematica Journal*, **8**(3), 2003 pp. 473–483.
- [20] J. Gurland, "Distribution of Definite and of Indefinite Quadratic Forms," *The Annals of Mathematical Statistics*, **26**, 1955 pp. 122–127.
- [21] M. L. Tiku, "Laguerre Series Forms of Non-central χ^2 and F Distributions," *Biometrika*, **52**(3), 1965 pp. 415–427.
- [22] N. L. Johnson and S. Kotz, *Distributions in Statistics: Continuous Univariate Distributions*, Vol. 2, New York: Wiley, 1970.
- [23] J. Durbin and G. S. Watson, "Testing for Serial Correlation in Least-Squares Regression II," *Biometrika*, **38**(1–2), 1951 pp. 159–178.
- [24] T. W. Anderson, *An Introduction to Multivariate Statistical Analysis*, 3rd ed., New York: John Wiley & Sons, 2003.
- [25] A. M. Mathai, "On the Distribution of the Likelihood Ratio Criterion for Testing Linear Hypotheses on Regression Coefficients," *Annals of the Institute of Statistical Mathematics*, **23**, 1971 pp. 181–197.
- [26] H. Cramér, "On Some Classes of Series Used in Mathematical Statistics," *Proceedings of the Sixth Scandinavian Congress of Mathematicians*, Copenhagen, 1925 pp. 399–425.

About the Author

Serge B. Provost is a professor of statistics in the Department of Statistical and Actuarial Sciences at The University of Western Ontario. He holds a Ph.D. in mathematics and statistics from McGill University. He is a fellow of the Royal Statistical Society and an associate of the Society of Actuaries. He co-authored two research monographs on quadratic and bilinear forms in random variables. His research interests include distribution theory, multivariate analysis, order statistics, time series, and computational statistics.

Serge B. Provost

Department of Statistical and Actuarial Sciences
The University of Western Ontario
London, Ontario, Canada N6A 5B7
sp@uwo.ca

Stochastic Integrals and Their Expectations

Wilfrid S. Kendall

This article explains how the *Itovsn3* package can be extended to add various properties and rules for `ItoIntegral`, which represents a stochastic or Itô integral. This allows us to introduce a further expectation operator and compute suitable expectations involving Itô integrals.

■ Introduction

This article describes the *Mathematica* package *ItoIntegralRules* that provides facilities to simplify and compute expectations of stochastic integrals. *ItoIntegralRules* extends the previous package *Itovsn3* [1, 2] that implements stochastic calculus within *Mathematica*.

Stochastic calculus is famous for providing the foundations for modern mathematical finance and is also used extensively in a large number of other areas of applied probability. The introductory text by Øksendal [3] strikes an excellent balance between theory and accessibility. Here we give a very brief review of the underlying concepts. A central notion for stochastic calculus is that of a (continuous) *semimartingale*: a random process X that can be written as the sum of a *local martingale* M (for example, Brownian motion) and a *drift process* V (a continuous process of locally bounded variation, typically the solution of some conventional differential equation). The decomposition $X = X(0) + M + V$ is unique and can be thought of as a decomposition of X into signal V plus noise M . Fundamental to the theory of stochastic calculus is *Itô's lemma*: if $f(X)$ is a smooth function of the semimartingale X , then

$$f(X) = f(X(0)) + \int f'(X) dX + \frac{1}{2} \int f''(X) d[X, X],$$

where $[X, X] = [M, M]$ is the quadratic variation, the unique nondecreasing process such that $M^2 - [M, M]$ is a local martingale begun at 0. In the case when M is Brownian motion, we find $[M, M] = t$. Care has to be taken when interpreting the integral $\int f'(X) dX$: nontrivial continuous local martingales M do not possess bounded variation, so the component $\int f'(X) dM$ of $\int f'(X) dX = \int f'(X) dM + \int f'(X) dV$ must be interpreted as a *stochastic* or *Itô integral*. (Since V is of locally bounded variation, the interpretation of $\int f'(X) dV$ is strictly classical.)

Itô's lemma, in conjunction with martingale theory, permits us to calculate effectively with semimartingales in *Mathematica*. In *Itovs3* [1, 2] the underlying algebra of stochastic calculus is implemented as an algebra of *stochastic differentials* dX , dM , and dV . This has facilitated several investigations into applied probability problems: examples given in [2] include explorations of the statistical theory of shape, coupling of diffusions, and computation of distributions of special random processes. The underlying principle of *Itovs3* is to recognize a second-order algebraic structure of differentials corresponding to the formula for Itô's lemma. Thus semimartingales X have stochastic differentials dX that can be multiplied together to obtain a differential measure of volatility of the semimartingale (for example, $dX^2 = d[X, X]$) and that possess drift parts that capture the underlying trend ($\text{Drift}[dX] = dV$ if $X = X(0) + M + V$).

The need to perform some calculations related to an image analysis problem [4, 5] supplied the initial motivation to extend *Itovs3* by adding the package *ItoIntegralRules* to more fully implement a notion of *Itô integral*, such as $\int f'(X) dM$ or $\int f'(X) dX$. Itô integrals $\int g(X) dX$ are represented in *Itovs3* using placeholders `ItoIntegral[g dM]` that possess the bare minimum of properties (loosely speaking, `ItoIntegral[dM] → M`). The new package *ItoIntegralRules* adds facilities to simplify expressions involving `ItoIntegral` in various ways and also adds an expectation operator $\mathbb{E}$. In particular, this allows us to address the calculations arising from the image analysis problem, which requires the derivation and further manipulation of formulas for means and variances for integrated Ornstein–Uhlenbeck processes. These specific calculations can of course be performed directly by hand; however, the computational framework provided by *ItoIntegralRules* covers a much wider range of possible calculations, so it should be of use elsewhere.

This article is divided into three sections: the first summarizes the issues of simplification of expressions involving `ItoIntegral`, the second introduces a notion of expectation and its interaction with `ItoIntegral`, and the conclusion discusses possibilities for further work.

□ Related Work

There are other implementations of stochastic calculus within a computer algebra package. Steele and Stine [6] adopt a diffusion-based approach, which has been developed further by Mark Fisher in the *ItosLemma.m* package [7]. Cyganowski [8] describes an approach using Maple, including a solver for stochastic differential equations.

□ Installation

The installation of *Itovs3* and *ItoIntegralRules* follows the usual procedure for *Mathematica* packages. Unpack the zip archive file *Itovs3.zip* (see Additional Material) either in the current working directory or in the Applications subdirectory of *Mathematica*'s AddOns directory (in the second case the packages will load no matter what is the current working directory). This will place the files

init.m (which contains the package *Itovs3*), *ItoIntegralRules.m*, and *ItoIntegralTests.nb* in the *Itovs3* subdirectory. The accompanying notebook, *ItoIntegralTests.nb*, contains detailed examples and unit tests for *ItoIntegralRules*.

After installation, *Itovs3* can be loaded and initialized and a single Brownian motion B can be introduced by

```
In[1]:= Needs["Itovs3`"];
        ItoReset[t, dt];
        BrownSingle[B, 0];
```

■ Simplification

ItoIntegralRules implements properties for *ItoIntegral*[dX] using an approach based on a family of simplification rules, exported by the package *ItoIntegralRules.m*. Rule names are prefixed by *ItoIntegral* or *ItoExpect* to avoid name clashes. The rule-based approach is preferred to canonical simplification techniques because, as a consequence of Itô's lemma, there will typically be inequivalent simplification strategies. For example, if B is Brownian motion, then Itô's lemma can be applied together with $[B, B] = t$ (or, in differential form, $dB^2 = dt$) to show

$$2 \int t B dB = t B^2 - \int B^2 dt - \int t dt,$$

and the preferred choice between the two equivalent forms will depend on context, in particular whether it is more convenient for expressions to contain stochastic or classical integrals. We now survey the major rules and briefly illustrate their use in simplification. More detailed information and unit tests can be found in *ItoIntegralTests.nb*.

□ Additivity and Linearity

ItoIntegralRules implements additivity: $\int (X + Y) dZ = \int X dZ + \int Y dZ$ is applied automatically once the package is loaded. We exemplify this by considering *ItoIntegral*[$a dB - b dB$], representing the stochastic integral $\int a dB - \int b dB$.

```
In[4]:= ItoIntegral[a dB - b dB]
Out[4]= ItoIntegral[a dB - b dB]

In[5]:= Needs["Itovs3`ItoIntegralRules`"]
In[6]:= ItoIntegral[a dB - b dB]
Out[6]= ItoIntegral[a dB] - ItoIntegral[b dB]
```

It would normally be convenient to extract constant coefficients a and b : *ItoIntegrationRules* defines *ItoIntegralExpandRule* to perform this. In this particular case, *Itovs3* can apply the original rules for *ItoIntegral* after linear expansion to deliver a complete solution.

In[7]:= ? ItoIntegralExpandRule

A rule to expand the argument of ItoIntegral,
pulling out factors which are non-random and non-constant.

In[8]:= SetAttributes[{a, b}, Constant];
ItoIntegral[a dB - b dB] /. ItoIntegralExpandRule

Out[9]= a B - b B

□ Relationship to Classical Integration

If the Itô integral involves no semimartingale terms other than the time term t (and its differential dt), then it can be rewritten as a classical time integral. *ItoIntegralRules* supplies *ItoIntegralClassicRule* to make this transformation and then the integral may possibly evaluate. (An implementation issue should be noted here. *ItoVsn3* is based on the total differentiation *Dt* operation, which assumes dependence unless explicitly stated otherwise. *Integrate* assumes symbols are constant by default. As long as the only quantities to vary in time are semimartingales, which would be the case in normal use of *ItoVsn3*, this presents no problems.)

In[10]:= ? ItoIntegralClassicRule

A rule which attempts to convert ItoIntegral into a classical integral.

In[11]:= ItoIntegral[t dt + B dB] /. ItoIntegralClassicRule

Out[11]= $\frac{t^2}{2} + \text{ItoIntegral}[B dB]$

□ A Simplification Strategy

These rules can be applied in a variety of ways. In important special cases their application can be systematized. Here is a simple example. Consider expressions formed from just one Brownian motion B , as defined earlier, and time t using only addition, multiplication, and (possibly iterated) integration with respect to B and t , which we shall call “polynomial semimartingales.” These can be reduced to expressions that involve classical integrals alone (no Itô integrals) by repeated application of specific formulas derived from stochastic integration by parts, itself derived from Itô’s lemma:

$$\int \left(\int p dt \right) q dB = \int p dt \int q dB - \int \left(\int q dB \right) p dt.$$

Taking into account the various structural variations (t , t^a , B , ... as in the lists 11, 12 following) for monomials p and q , there are 80 different rules to be considered! It is therefore convenient (and more reliable) to construct the various resulting rules automatically as follows (the rule set is also *tested* automatically in the accompanying notebook *ItoIntegralTests.nb*).

```

In[12]:= ItoIntegralRewriteRuleset =
Module[{a, b, c, d, conv, l1, l2},
SetAttributes[{a, b, c, d, conv}, Constant];
conv = Map[(# → Condition[Pattern[#, Blank[]], Greater[#, -1]]) &,
{a, b, c, d}];
l1 = {t, ta, B, Bb, t B, t Bb, ta B, ta Bb};
l2 = {t, tc, B, Bd, t B, t Bd, tc B, tc Bd};
Map[(#[[1]] /. conv) → #[[2]]] &, Flatten[
{
Map[Solve[
# ItoIntegral[dB] == (ItoIntegral[ItoD[# ItoIntegral[dB]]] /.
ItoIntegralExpandRule), ItoIntegral[# dB]] &, l1],

Map[Solve[ItoIntegral[# dt] ItoIntegral[dB] ==
(ItoIntegral[ItoD[ItoIntegral[# dt] ItoIntegral[dB]]] /.
ItoIntegralExpandRule),
ItoIntegral[ItoIntegral[# dt] dB]] &, l2],

Map[Solve[ItoIntegral#[[2]] dt ItoIntegral#[[1]] dB ==
(ItoIntegral[ItoD[ItoIntegral#[[2]] dt]
ItoIntegral#[[1]] dB]] /. ItoIntegralExpandRule),
ItoIntegral[ItoIntegral#[[2]] dt] #[[1]] dB]] &,
Outer[List, l1, l2], {2}]
}
]
]
];

```

The rule set must be applied iteratively to suitable expressions until they stop changing, so we use `FixedPoint` to construct an appropriate function. (Note the argument `iter`, controlling maximum number of iterations, is set by default to `Infinity` since there is no *a priori* upper bound on the number of iterations required to simplify a general polynomial semimartingale.)

```

In[13]:= applyItoRewriteRules[x_, iter_: Infinity] :=
FixedPoint[(# /. ItoIntegralExpandRule /. ItoIntegralRewriteRuleset /.
ItoIntegralClassicRule // Expand) &, x, iter]

```

We can test this simplification procedure on a famous result from stochastic calculus: the family of Hermite polynomials forms a structure that is preserved by Itô integration.

```

In[14]:= H[k_, x_, t_] := (2 t)k/2 HermiteH[k,  $\frac{x}{\sqrt{2 t}}$ ] // Expand;

```

With this definition, we have

$$\int H[n, B, t] dB = \frac{H[n+1, B, t]}{2(n+1)}$$

and here we test this for the first 10 values of n .

```

In[15]:= Table[
  Expand[ $\frac{H[n+1, B, t]}{2 (n+1)}$ ] ==
  applyItoRewriteRules[ItoIntegral[H[n, B, t] dB]],
  {n, 1, 10}]

Out[15]= {True, True, True, True, True, True, True, True, True, True}

```

Variations on this approach can be devised for polynomial semimartingales based on time t and n independent Brownian motions: see Gaines [9, 10] who applies the notion of Lyndon bases for shuffle products on free algebras. Rather than pursuing this, we now turn to consider expectations of stochastic integrals.

■ Expectation

Itô's lemma may be employed in computation of expectations of semimartingale expressions as follows. If we wish to evaluate $\mathbb{E}[f(X)]$, then we may expand $f(X)$ using Itô's lemma. For well-behaved f (e.g., polynomial functions of Brownian motion), we may replace the differential dX in the expectation of the stochastic integral $\mathbb{E}[\int f'(X) dX]$ by its drift dV to obtain $\mathbb{E}[\int f'(X) dV]$. If the drift is zero (as is the case for Brownian motion), the integral then vanishes; if the drift is deterministic (e.g., the time process t), then we may simplify further to obtain $\int \mathbb{E}[f'(X)] dV$. It follows by induction that we can evaluate an expectation completely if the semimartingale expression X is a combination of linear operations, multiplication, and stochastic integration performed on time t and n independent Brownian motions (what we called a polynomial semimartingale in the previous section).

ItoIntegralRules therefore defines an expectation operator $\mathbb{E}$, which possesses basic linearity properties, and an associated function $\mathbb{EE}$, which applies transformations of the previous form whenever the semimartingale expression is a polynomial semimartingale.

□ Examples

We first consider some simple examples of expectations of polynomial semimartingales. Here is a computation of $\mathbb{E}[(\int B^4 dB)^2]$.

```

In[16]:= E[ItoIntegral[B^4 dB]^2]

Out[16]= E[ItoIntegral[B^4 dB]^2]

In[17]:= E[ItoIntegral[B^4 dB]^2] /. E -> EE

Out[17]= 21 t^5

```

Higher-order powers can be dealt with in an equally direct manner, though with increasing computational effort. Here we tabulate $\mathbb{E}[(1 + \int B^4 dB)^n]$ for values of n up to 5.

```
In[18]:= Table[EE[(1 + ItoIntegral[B^4 dB])^n], {n, 1, 5}] // TableForm
Out[18]//TableForm=
1
1 + 21 t^5
1 + 63 t^5
1 + 126 t^5 +  $\frac{2967183 t^{10}}{5}$ 
1 + 210 t^5 + 2967183 t^{10}
```

Iterated integrals can be disposed of in a similar fashion. Here we evaluate $\mathbb{E}[(\int e^t \int e^{-t} dB dt)^8]$. Note that $\mathbb{E}$ can deal with nonpolynomial functions of t .

```
In[19]:= EE[ItoIntegral[dt e^t ItoIntegral[dB e^{-t}]]^8]
Out[19]=  $\frac{105}{16} (3 - 4 e^t + e^{2t} + 2 t)^4$ 
```

Here we evaluate $\mathbb{E}[(\int (\int B \sin(t) dt)^2 dB)^2]$.

```
In[20]:= EE[ItoIntegral[ItoIntegral[B Sin[t] dt]^2 dB]^2]
Out[20]=  $\frac{3}{256} (72 t + 96 t^3 + 320 t \cos[2 t] + 28 t \cos[4 t] -$   

 $160 \sin[2 t] + 128 t^2 \sin[2 t] - 25 \sin[4 t] + 8 t^2 \sin[4 t])$ 
```

Nonpolynomial semimartingales are left unsimplified if the nonpolynomial part involves Brownian motions, as in this evaluation of $\mathbb{E}[(B + \sin(B))^4]$.

```
In[21]:= EE[(B + Sin[B])^4]
Out[21]=  $3 t^2 + 4 \mathbb{E}[B^3 \sin[B]] + 6 \mathbb{E}[B^2 \sin[B]^2] + 4 \mathbb{E}[B \sin[B]^3] + \mathbb{E}[\sin[B]^4]$ 
```

There are further techniques available for dealing with nonpolynomial semimartingales. For example, consider the evaluation of $\mathbb{E}[\sin(B)^2]$. We could of course evaluate this directly using the density for the random variable B , and this itself can be automated using *mathStatica* [11]. However, we can also make progress using two further *ItoIntegralRules* (*ItoExpectItoIntegralRule*, *ItoExpectExpandRule*), which are components of $\mathbb{E}$. The first of these implements the interplay between expectation and drift described at the start of this section, while the second expands $\mathbb{E}$ linearly and extracts nonrandom terms.

```
In[22]:= With[{X = Sin[B]^2},
  EE[X] /. ItoExpectItoIntegralRule /. Cos[x_]^2 -> 1 - Sin[x]^2 /.
  ItoIntegralExpandRule /. ItoExpectExpandRule]
Out[22]=  $t - 2 \text{ItoIntegral}[dt \mathbb{E}[\sin[B]^2]]$ 
```

We have thus obtained a recursive expression for $\mathbb{E}[\sin(B)^2]$ that can now be used to form a differential equation by further developing this code.

```

In[23]:= With[{X = Sin[B]^2},
  DSolve[{D[x[t], t] == D[EE[X] /. ItoExpectItoIntegralRule /.
    Cos[x_]^2 -> 1 - Sin[x]^2 /. ItoIntegralExpandRule /.
    ItoExpectExpandRule /. E[X] -> x[t] /.
    ItoIntegralClassicRule, t],
  x[0] == EE[InitialValue[0, X]], x[t], t]]
Out[23]= {{x[t] -> 1/2 e^{-2 t} (-1 + e^{2 t})}}

```

So we obtain

$$\mathbb{E}[\sin(B)^2] = \frac{1}{2} (1 - e^{-2t}).$$

Further examples can be found in `ItoIntegralTests.nb`. This differential equation approach can be applied even when we require the expectation of a quantity that is not a simple function of Brownian motion. See, for example, the treatment of the distribution of the stochastic area integral $\int A dB - \int B dA$ in [2].

□ Calculations for the Ornstein–Uhlenbeck Process

The original motivation for this work was to provide an environment to aid computations of expectations of quantities associated with the integrated Ornstein–Uhlenbeck process. To illustrate this, we use a pair of stochastic differential equations

$$\begin{aligned} dU &= dB - \frac{\alpha}{2} U dt \\ dX &= U dt \end{aligned}$$

to define an Ornstein–Uhlenbeck process and its integrated variant. To do this in *Itovs3* we use `Itosde`.

```

In[24]:= SetAttributes[{α, U0, X0}, Constant];
Itosde[U, dU == dB - α/2 U dt, U0];
Itosde[X, dX == U dt, X0];

```

Note that it must be stated explicitly that both α and the initial values $U0$, $X0$ are `Constant` in time. It is possible to solve these linear stochastic differential equations in closed form: $U = e^{-\alpha t/2} (U(0) + \int e^{\alpha t/2} dB)$ and $X = X(0) + \int U dt$. We express the solutions using `ItoIntegral`:

```

In[27]:= UU = e^{-α t/2} (U0 + ItoIntegral[e^{α t/2} dB])
Out[27]= e^{-t α/2} (U0 + ItoIntegral[dB e^{t α/2}])
and
In[28]:= XX = X0 + ItoIntegral[U dt] /. U -> UU
Out[28]= X0 + ItoIntegral[dt e^{-t α/2} (U0 + ItoIntegral[dB e^{t α/2}])]

```

and verify directly that they satisfy the relevant stochastic differential equations:

```
In[29]:= ItoD[UU] == (dB -  $\frac{\alpha}{2}$  UU dt && InitialValue[0, UU] == U0) // Simplify
```

```
Out[29]= True
```

```
In[30]:= Simplify[ItoD[XX] == UU dt && InitialValue[0, XX] == X0]
```

```
Out[30]= True
```

We now compute mean values

```
In[31]:= EE[UU]
```

```
Out[31]=  $e^{-\frac{t\alpha}{2}}$  U0
```

```
In[32]:= EE[XX] // Expand
```

```
Out[32]=  $X0 + \frac{2U0}{\alpha} - \frac{2e^{-\frac{t\alpha}{2}}U0}{\alpha}$ 
```

and the variance-covariance matrix (using $\text{Cov}[X, Y] = \mathbb{E}\mathbb{E}[X, Y] - \mathbb{E}\mathbb{E}[X]\mathbb{E}\mathbb{E}[Y]$).

```
In[33]:= Outer[Cov, {B, UU, XX}, {B, UU, XX}] // MatrixForm
```

```
Out[33]//MatrixForm=
```

$$\begin{pmatrix} t & \frac{2}{\alpha} - \frac{2e^{-\frac{t\alpha}{2}}}{\alpha} & -\frac{4}{\alpha^2} + \frac{4e^{-\frac{t\alpha}{2}}}{\alpha^2} + \frac{2t}{\alpha} \\ \frac{2}{\alpha} - \frac{2e^{-\frac{t\alpha}{2}}}{\alpha} & \frac{1}{\alpha} - \frac{e^{-t\alpha}}{\alpha} & \frac{2}{\alpha^2} + \frac{2e^{-t\alpha}}{\alpha^2} - \frac{4e^{-\frac{t\alpha}{2}}}{\alpha^2} \\ -\frac{4}{\alpha^2} + \frac{4e^{-\frac{t\alpha}{2}}}{\alpha^2} + \frac{2t}{\alpha} & \frac{2}{\alpha^2} + \frac{2e^{-t\alpha}}{\alpha^2} - \frac{4e^{-\frac{t\alpha}{2}}}{\alpha^2} & -\frac{12}{\alpha^3} - \frac{4e^{-t\alpha}}{\alpha^3} + \frac{16e^{-\frac{t\alpha}{2}}}{\alpha^3} + \frac{4t}{\alpha^2} \end{pmatrix}$$

Computation of the fourth central moment is equally direct.

```
In[34]:= EE[(XX - EE[XX])^4]
```

```
Out[34]=  $\frac{48e^{-2t\alpha} \left( -1 + 4e^{\frac{t\alpha}{2}} + e^{t\alpha} (-3 + t\alpha) \right)^2}{\alpha^6}$ 
```

The image analysis application required the derivation of the conditional distribution of X and U at a specified time s given the values of X and U at 0 and t , for $0 < s < t$. Since (X, U) is a Gaussian process, we can find this by straightforward use of the `Statistics`MultinormalDistribution`` package once the means and variance-covariance matrix are calculated for the various values of X and U at times 0, s , and t . Of course it is possible to derive the conditional distribution by hand; an advantage of working in *Mathematica* is that we are then able to proceed directly to simulation experiments and so forth.

■ Conclusion

We have shown in this article (and in `ItoIntegralTests.nb`) how the package *ItoIsm3* can be extended to provide the ability to manipulate stochastic integrals and compute expectations of stochastic calculus quantities. Further extensions are possible—we have already noted Gaines' work [9, 10] on simplification of

polynomial semimartingales based on several independent Brownian motions. A more ambitious extension would be to generalize and automate the differential equation strategy described here for computing expectations, such as $\mathbb{E}[\sin(B)^2]$. This would be a demanding project involving the automatic recognition of differential equations satisfied by the expectation in question. We have also noted the possibility of interaction with other packages such as *mathStatistica*. For example, Colin Rose has pointed out how the definition of $\mathbb{E}$ can be modified so as to invoke *mathStatistica*'s Expect function when the expression has been determined not to be a polynomial semimartingale. This would provide a direct means of computing $\mathbb{E}[\sin(B)^2]$; however, differential equation techniques would still be required when dealing with expressions involving stochastic integrals.

■ Acknowledgments

This work was supported by EPSRC grant GR/M75785. I am grateful for many helpful discussions with my collaborators on this grant: Abhir Bhalerao, Elke Thönnies, and Roland Wilson. It is a pleasure to acknowledge the hospitality of David Nualart and the Statistics Department of the University of Barcelona during part of the writing process. Thanks are also due to the insightful remarks of two referees, and I am particularly indebted to Colin Rose for his enthusiastic editorial contributions, which ranged from incisive stylistic advice all the way to source-code optimization!

Calculations were carried out on a Sun Ultra-10 workstation using *Mathematica* 4.1.

■ References

- [1] W. S. Kendall, "Itôvsn3: Doing Stochastic Calculus with *Mathematica*," *Economic and Financial Modeling with Mathematica* (H. R. Varian, ed.), New York: Springer-Verlag, 1993 pp. 214–239.
- [2] W. S. Kendall, "Stochastic Calculus in *Mathematica*: Software and Examples," *University of Warwick Department of Statistics Research Report 333* (Jan 2003) www.warwick.ac.uk/go/wsk/ppt/333.pdf.
- [3] B. K. Øksendal, *Stochastic Differential Equations*, 5th ed., Berlin: Springer Universitext, 1998.
- [4] A. Bhalerao, E. Thönnies, W. S. Kendall, and R. G. Wilson, "Inferring Vascular Structure from 2D and 3D Imagery," in *Medical Image Computing and Computer-Assisted Intervention, Proceedings MICCAI 2001*, (W. J. Niessen and M. A. Viergever, eds.), pp. 820–828; *Lecture Notes in Computer Science 2208*, Springer, 2001 pp. 820–828; *University of Warwick Department of Statistics Research Report 392* (Jan 2002) www.warwick.ac.uk/go/wsk/ppt/392.pdf.

- [5] E. Thönnies, A. Bhalerao, W. S. Kendall, and R. G. Wilson, "A Bayesian Approach to Inferring Vascular Tree Structure from 2D Imagery," in *International Conference on Image Processing, Proceedings ICIP 2002*, (W. J. Niessen and M. A. Viergever, eds.), pp. 937–939; *University of Warwick Department of Statistics Research Report 391* (Oct 2001) www.warwick.ac.uk/go/wsk/ppt/391.pdf.
- [6] J. M. Steele and R. A. Stine, "Mathematica and Diffusions," *Economic and Financial Modeling with Mathematica*, (H. R. Varian, ed.), New York: Springer-Verlag, 1993 pp. 192–214.
- [7] M. Fisher, "ItosLemma.m" *MathSource* (May 20, 2002) library.wolfram.com/database/MathSource/1170 (or www.markfisher.net/~mefisher/mma/mathematica.html).
- [8] S. Cyganowski, "Solving Stochastic Differential Equations with Maple" *Computer Algebra: Maple Tech*, **3**(2), 1986 pp. 35–40.
- [9] J. G. Gaines, "The Algebra of Iterated Stochastic Integrals," *Stochastics and Stochastics Reports*, **49**(3–4), 1994 pp. 169–179.
- [10] J. G. Gaines, "A Basis for Iterated Stochastic Integrals," *Mathematics and Computers in Simulation* [online], **38**(1–3), 1995 pp. 7–11. [dx.doi.org/10.1016/0378-4754\(93\)E0061-9](https://doi.org/10.1016/0378-4754(93)E0061-9).
- [11] C. Rose and M. D. Smith, *Mathematical Statistics with Mathematica*, Springer Texts in Statistics, New York: Springer-Verlag, 2002.

■ Additional Material

ItoVsn3.zip (zip archive containing the files `init.m`, `ItoIntegralRules.m`, and `ItoIntegralTests.nb`).

Available at www.mathematica-journal.com/issue/v9i4/download.

About the Author

Wilfrid S. Kendall is a professor of statistics at the University of Warwick, with varied research interests primarily concerning the way computing and geometry influence probability and statistics.

Wilfrid S. Kendall

Department of Statistics
University of Warwick
Coventry CV4 7AL, UK
w.s.kendall@warwick.ac.uk
www.warwick.ac.uk/go/wsk

Bootstrap Tutorial

Hal Varian

This article is an elementary tutorial describing how to implement the bootstrap method in *Mathematica*.

■ Introduction

According to Efron and Tibshirani [1], “The bootstrap is a computer-based method for assigning measures of accuracy to sample estimates.” This remarkable computational technique allows us to generate an estimate of the sampling distribution of almost any statistic using very simple methods. There are several books and articles that explain the statistical theory behind this method, including [1], which provides an excellent introduction. There are also two notebooks available on *MathSource* that implement the bootstrap in *Mathematica* [2, 3].

■ Load Required Packages

```
In[1]:= << Statistics`DescriptiveStatistics`  
        << Statistics`DataManipulation`  
        << Graphics`Graphics`  
        << Statistics`LinearRegression`
```

To make debugging easier, we will seed the random number generator so we get the same random numbers each time the notebook is executed.

```
In[5]:= SeedRandom[123456]
```

■ Distribution of Sample Mean

We begin by reviewing the most elementary problem in statistics, the distribution of the sample mean.

Let (x_i) for $i = 1, \dots, n$ be a sample of IID random variables with $E(x_i) = \mu$ and $\text{var}(x_i) = \sigma^2$. The sample mean is defined by

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i.$$

Since the observations are random variables, the sample mean is a random variable and we can, in principle, compute its distribution. Using the properties of expected value, it is a standard exercise to show that $E(\bar{x}) = \mu$ and

$\text{var}(\bar{x}) = \sigma^2 / n$, which gives us the first two moments of the sampling distribution. If the x_i terms are distributed $N(\mu, \sigma^2)$, we know that a sum of these variables will be normally distributed, so $\bar{x}$ will be distributed $N(\mu, \sigma^2 / n)$.

But what do we do if the x_i s are not normally distributed or if we are computing some function of the sample that is more complex than the mean?

Classical statistics was driven by analytic tractability, and the methods used in classical statistics only apply to certain well-behaved distributions and certain, mostly linear, computations. With modern computers, analytic complexity is no barrier to computing estimates of the sampling distribution of almost any statistic, as we demonstrate next using Monte Carlo simulation.

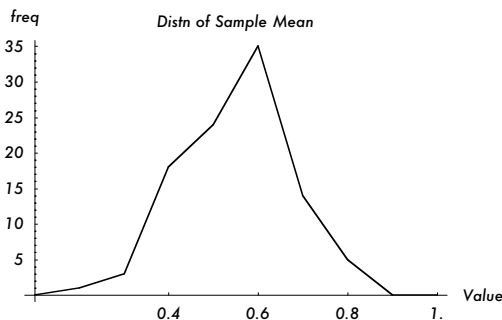
Here we draw a list of 25 uniformly distributed random numbers, compute the mean, and repeat this 100 times. This will give us 100 different estimates of the mean of the underlying distribution.

```
In[6]:= sampleList = Table[Mean[Table[Random[], {25}]], {100}];
```

Let us look at the distribution of these 100 calculated means; this frequency distribution can be viewed as an estimate of the true sampling distribution.

```
In[7]:= Print["Sample list mean: ", Mean[sampleList],  
" and variance: ", Variance[sampleList]]  
ticks1 = {{1, .1}, {4, .4}, {6, .6}, {8, .8}, {10, 1.0}}, Automatic};  
ListPlot[BinCounts[sampleList, {.25, .75, .05}],  
PlotJoined -> True, AxesLabel -> {"Value", "freq"},  
PlotLabel -> "Distn of Sample Mean", Ticks -> ticks1]
```

Sample list mean: 0.501391 and variance: 0.00361008



Since the underlying random variable is uniformly distributed on $[0,1]$, the estimated mean should be close to 0.5. The variance of the uniform distribution is

$$\text{theoryVar} = N \left[\int_0^1 \left(x - \frac{1}{2} \right)^2 dx \right]$$

```
Out[10]= 0.0833333
```

So the variance of the sample mean of 25 observations should be

```
In[11]:= meanVar =  $\frac{\text{theoryVar}}{25}$ 
```

```
Out[11]= 0.00333333
```

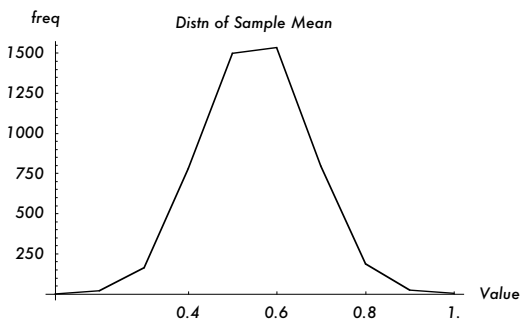
The estimates we have computed should not be too far from these numbers.

We can do the same thing for 5000 repetitions, in which case the estimated results should be much closer to the theoretical predictions.

```
In[12]:= sampleList = Table[Mean[Table[Random[], {25}]], {5000}];

In[13]:= Print["Sample list mean: ", Mean[sampleList],
  " and variance: ", Variance[sampleList]]
ticks1 = {{1, .1}, {4, .4}, {6, .6}, {8, .8}, {10, 1.0}}, Automatic];
ListPlot[BinCounts[sampleList, {.25, .75, .05}],
  PlotJoined → True, AxesLabel → {"Value", "freq"},
  PlotLabel → "Distn of Sample Mean", Ticks → ticks1]
```

Sample list mean: 0.501418 and variance: 0.00330898



The Monte Carlo method can be used to compute an estimate of the sampling distribution for virtually any statistic, as long as we know the distribution from which the samples are drawn.

■ The Bootstrap

Unfortunately, in most cases we do not know the underlying distribution from which the sample is drawn. At best we may suspect that the true distribution is in some family of distributions, but we generally do not know the parameters of the distribution. If we did, we could just apply the theoretical formulas and be done with it.

So suppose that we have just one sample. Is there any way to use that one sample to compute an estimate of the sampling distribution of a statistic? This is where the bootstrap comes in.

The idea is to repeatedly sample (with replacement) from the single sample you have, and use these “samples” to compute the sampling distribution of the

statistic in which you are interested. In the previous Monte Carlo exercise, we drew a “fresh” sample each time; in the bootstrap case, we *resample* from the single sample that we have. Other than that difference, the procedures are essentially the same. If our original sample is reasonably representative of the population, then resampling from that sample should look pretty much like drawing a new sample.

The remarkable thing about the bootstrap is that even though we only have a single sample, it can often be used to give a quite good estimate of what would happen if we really were able to draw new, fresh samples.

Here is a function that will resample (with replacement) from a list.

```
In[16]:= Resample[list_] :=
         list[[Table[Random[Integer, {1, Length[list]}], {Length[list]}]]
```

To make sure it works, we will try it on a test list.

```
In[17]:= test = {a, b, c, d, e, f, g}
Out[17]= {a, b, c, d, e, f, g}
In[18]:= Resample[test]
Out[18]= {f, c, d, b, e, f, c}
```

Here is a fixed random sample of 25 numbers.

```
In[19]:= SeedRandom[123456]
In[20]:= theSample = Table[Random[], {25}];
```

Now we will resample from this fixed sample 5000 times and look at this distribution of estimated means.

```
In[21]:= bootList = Table[Mean[Resample[theSample]], {5000}];
```

■ Comparison of the Distributions

Here are some summary statistics comparing the sampling distribution computed earlier and the bootstrap distribution.

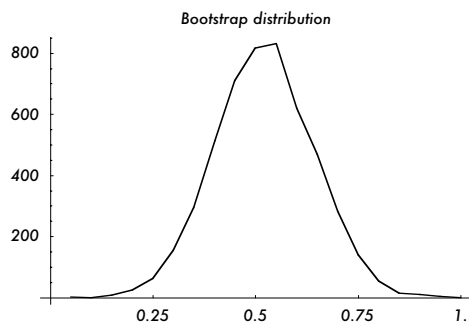
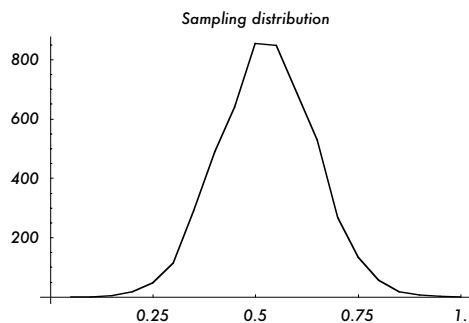
```

In[22]:= Print["Sample list mean: ", Mean[sampleList],
  " and variance: ", Variance[sampleList]]
Print["Boot list mean: ", Mean[bootList],
  " and variance: ", Variance[bootList]]
ticks2 = {{5, .25}, {10, .5}, {15, .75}, {20, 1.0}}, Automatic};
ListPlot[BinCounts[sampleList, {.25, .75, .025}], PlotJoined → True,
  PlotLabel → "Sampling distribution", Ticks → ticks2]
ListPlot[BinCounts[bootList, {.25, .75, .025}], PlotJoined → True,
  PlotLabel → "Bootstrap distribution", Ticks → ticks2]

```

Sample list mean: 0.501418 and variance: 0.00330898

Boot list mean: 0.497918 and variance: 0.00356321



Note that the distributions are reasonably similar. This is the genius of the bootstrap: resampling from the single sample provides a reasonable way to estimate what would happen if we actually drew many separate samples.

■ Bootstrapping Other Statistics

We can generalize the previous operations to bootstrap other statistics. Here is a function that will bootstrap any statistic `stat` of any sample.

```

In[27]:= bootIt[sample_, n_, stat_] := Table[stat[Resample[sample]], {n}]

```

```

In[28]:= bootIt[theSample, 10, Mean]

Out[28]= {0.420568, 0.628813, 0.450471, 0.478979, 0.543052,
          0.522497, 0.490412, 0.493006, 0.425277, 0.60749}

In[29]:= bootIt[theSample, 10, StandardDeviation]

Out[29]= {0.285901, 0.24937, 0.271834, 0.220513, 0.315868,
          0.308899, 0.33619, 0.302769, 0.319604, 0.294518}

In[30]:= bootIt[theSample, 10, Median]

Out[30]= {0.395873, 0.395873, 0.395873, 0.642715, 0.591733,
          0.60591, 0.591733, 0.322472, 0.322472, 0.306926}

```

The sampling distribution of the mean and standard deviation have been well studied. The sampling distribution of the median, on the other hand, is less well known, but it is trivial to compute an estimate of it with the bootstrap.

■ Regression

Now we will bootstrap a regression model. There are two ways to do this: we can assume that the matrix of explanatory variables is fixed or random. The fixed case is easier on both theoretical and computational grounds, but the random case is more realistic in most applications of interest. To keep the exposition simple, we will only examine the fixed regressors case. This case is applicable if, for example, the explanatory variables were created as part of a controlled experiment (see [1] for a more detailed discussion).

We start with some data and estimate the regression model.

```

In[31]:= SeedRandom[123456]

In[32]:= xData = Table[Random[], {50}];
          yData = Table[Random[], {50}];

In[34]:= Regress[Transpose[{xData, yData}], {1, x}, x]

Out[34]= {ParameterTable →
          Estimate   SE           TStat      PValue
          1   0.463401  0.0803805  5.7651     5.71645 × 10-7,
          x   0.0606434  0.13231   0.458344   0.648773
          RSquared → 0.00435759, AdjustedRSquared → -0.016385,
          EstimatedVariance → 0.0805626,
          ANOVATable →
          DF   SumOfSq   MeanSq   FRatio   PValue
          Model 1   0.0169246  0.0169246  0.21008  0.648773 }
          Error 48  3.867      0.0805626
          Total 49  3.88393

```

Since the x and the y variables are IID, the intercept term should be the mean of the y variable (.5) and the coefficient on the x variable should be about 0. In this

case the coefficient on the x variable is not significantly different from 0, as illustrated by the low value of the t -statistic.

To do the bootstrap, we form a set of the data pairs and then draw a random sample, with replacement, from that set. The same resampling function we used before works for this.

```
In[35]:= testData = Transpose[{{a1, a2, a3}, {b1, b2, b3}}]
Out[35]= {{a1, b1}, {a2, b2}, {a3, b3}}
In[36]:= Resample[testData]
Out[36]= {{a3, b3}, {a3, b3}, {a3, b3}}
```

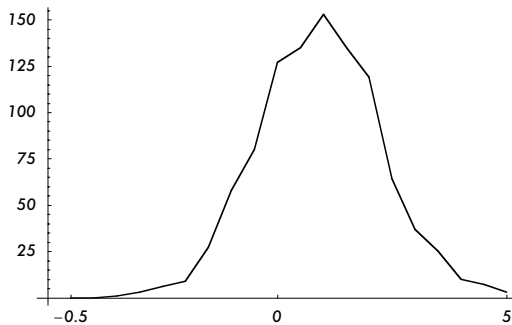
Now we simply run our regression on the resampled data and keep track of the regression coefficient. Here is a function that runs a regression and returns the slope coefficient.

```
In[37]:= coeff[data_] := Rest[Fit[data, {1, xx}, xx]][[1]]
In[38]:= allData = Transpose[{xData, yData}];
In[39]:= coeff[allData]
Out[39]= 0.0606434
```

Now we will do the bootstrap on the coefficient 1000 times.

```
In[40]:= slopeBoot = Table[coeff[Resample[allData]], {1000}];
In[41]:= Print["Boot list mean: ", Mean[slopeBoot],
  " and standard deviation: ", StandardDeviation[slopeBoot]]
ListPlot[BinCounts[slopeBoot, {- .5, .5, .05}], PlotJoined → True,
  Ticks → {{1, - .5}, {10, 0}, {20, +5}}, Automatic]
```

Boot list mean: 0.0674443 and standard deviation: 0.134648



The mean of the sampling distribution should be close to zero, and its standard deviation should be about what the regression computed for the standard error of the coefficient.

The bootstrap is a handy tool and particularly easy to implement in *Mathematica*. With the resampling functions described earlier, we can estimate the sampling distribution of virtually any statistic.

■ References

- [1] B. Efron and R. J. Tibshirani, *An Introduction to the Bootstrap*, Boca Raton, FL: CRC Press, 1994.
- [2] S. Aksenov, "Confidence Intervals by Bootstrap," *MathSource*. (Jan 27, 2003) library.wolfram.com/infocenter/MathSource/4272.
- [3] E. Siniksaran, "BootStrapPackage: A Package of Bootstrap Algorithms for Mean, Simple Linear Regression Models and Correlation Coefficient," *MathSource*. (May 30, 2002) library.wolfram.com/infocenter/MathSource/815.

About the Author

Hal Varian is the Class of 1944 Professor in the School of Information Management and Systems, the Haas School of Business, and the Department of Economics at UC Berkeley. He has edited two books on economic and financial applications of *Mathematica*.

Hal Varian

www.sims.berkeley.edu/~hal

Stable Distributions in Mathematica

Robert H. Rimmer
John P. Nolan

A *Mathematica* package for the computation of stable distributions is demonstrated. Calculations of stable density, distribution, and quantile functions can be performed and stable random variables generated. Data can be fit to stable parameters by the maximum likelihood method. Extreme precision calculations are possible. The *StableDistribution* package and documentation are provided in the Additional Material section.

■ Introduction

Stable distributions are a rich class of probability distributions that allow skewness and heavy tails and have many intriguing mathematical properties. The class was characterized by Paul Levy in his study of sums of independent, identically distributed terms in the 1920s [1]. Stable distributions have been proposed as models for many types of physical and economic systems. There are solid theoretical reasons for expecting a non-Gaussian stable model, for example, reflection off a rotating mirror yielding a Cauchy distribution, hitting times for a Brownian motion yielding a Levy distribution, and the gravitational field of stars yielding the Holtsmark distribution. The generalized central limit theorem states that the only possible nontrivial limit of normalized sums of independent, identically distributed terms is stable. It is argued that some observed quantities are the sum of many small terms; hence, a stable model should be used to describe such systems. For a detailed exposition of stable distributions and their applications, see [2].

The lack of formulae for densities and distribution functions for the full parameter range has limited the use of stable distributions by practitioners. Recently high-speed processors have made numerical calculations of stable functions feasible, and interest in their use has been increasing, especially in the study of financial markets based on Mandelbrot's work [3, 4, 5]. Rose and Smith [6] have used *Mathematica* to generate special functions to calculate stable densities and cumulative distributions, but these special functions exist only for certain rational values of stable parameters. Our program takes advantage of the Zolotarev integral representation of the stable density and the capabilities of the function, `NIntegrate`, to accurately calculate stable distributions for nearly the whole

parameter range. Its availability in *Mathematica* should make stable distributions accessible to a wide audience across all commonly used computing environments.

■ Parameterizations of Stable Laws

A general stable distribution requires four parameters to describe it:

- index of stability or characteristic exponent $\alpha \in (0, 2]$
- skewness parameter $\beta \in [-1, 1]$
- scale parameter $\gamma > 0$
- location parameter $\delta \in \mathbb{R}$

There are multiple parameterizations for stable laws, and much confusion has been caused by these different parameterizations. This package includes two parameterizations. The characteristic functions in the $S(\alpha, \beta, \gamma, \delta; 0)$ and $S(\alpha, \beta, \gamma, \delta; 1)$ parameterizations follow.

$S(\alpha, \beta, \gamma, \delta; 0)$:

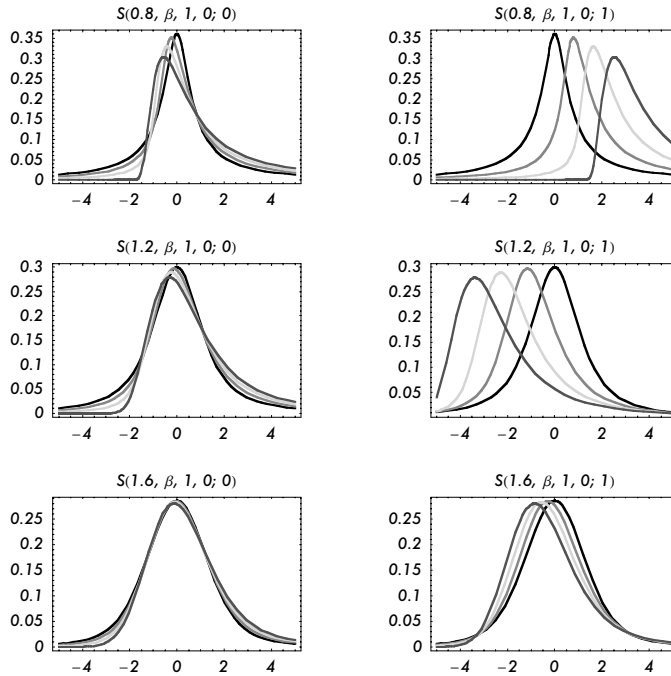
$$\begin{aligned} \exp\left(i u \delta - \gamma^\alpha \text{Abs}[u]^\alpha \left(1 + i \beta (-1 + \text{Abs}[u \gamma]^{1-\alpha}) \text{Sign}[u] \tan\left[\frac{\pi \alpha}{2}\right]\right)\right) & \quad \alpha \neq 1 \\ \exp\left(i u \delta - \frac{\gamma \text{Abs}[u] (\pi + 2 i \beta \text{Log}[\text{Abs}[u \gamma]] \text{Sign}[u])}{\pi}\right) & \quad \alpha = 1 \end{aligned}$$

and

$S(\alpha, \beta, \gamma, \delta; 1)$:

$$\begin{aligned} \exp\left(i u \delta - \gamma^\alpha \text{Abs}[u]^\alpha \left(1 - i \beta \text{Sign}[u] \tan\left[\frac{\pi \alpha}{2}\right]\right)\right) & \quad \alpha \neq 1 \\ \exp\left(i u \delta - \gamma \text{Abs}[u] \left(1 + \frac{2 i \beta \text{Log}[\text{Abs}[u]] \text{Sign}[u]}{\pi}\right)\right) & \quad \alpha = 1 \end{aligned}$$

The $S(\alpha, \beta, \gamma, \delta; 0)$ parameterization is the best choice for numerical computation of stable distributions, since it has the simplest form for the characteristic function that is continuous in all four parameters. The $S(\alpha, \beta, \gamma, \delta; 1)$ parameterization has been commonly used in economic literature and has the property that the location parameter, δ , is the mean when $\alpha > 1$. The following graphics array demonstrates these two parameterizations. The scale parameter, γ , has been set to 1 and the location parameter, δ , is 0. Each graph shows varying values of β (0, 1/3, 2/3, 1 in black, red, green, and blue colors, respectively) at the value of α and parameterization type listed on the graph. Negative values of β would give mirror image graphs. When α is near 1, the mode in the $S(\alpha, \beta, \gamma, \delta; 1)$ parameterization moves dramatically; the $S(\alpha, \beta, \gamma, \delta; 0)$ parameterization progressively reduces this motion as α approaches 1.



■ Using the Package

The *StableDistribution* package provides several basic functions needed for the calculation of stable distributions. They are all designed with identical input, output, and default formats. The package comes with a Help Browser that explains in more detail how each function works. It also includes a few more functions that are not described in this paper. The package allows both the 0 and 1 parameterizations for all the following functions, but in the examples and discussion only $S(\alpha, \beta, \gamma, \delta; 1)$ will be used.

Place the unzipped folder, *StableDistribution* (see Additional Material), in the *Mathematica* applications folder. After rebuilding the Help Browser index, documentation will be available in the Help Browser Add-ons & Links category. The following command loads the program.

```
In[1]:= << StableDistribution`SMath`
```

SPDF[x, { $\alpha, \beta, \gamma, \delta$ }, param] calculates density of a stable $S(\alpha, \beta, \gamma, \delta; \text{param})$ distribution at the value in x.

SCDF[x, { $\alpha, \beta, \gamma, \delta$ }, param] calculates cumulative distribution function of a stable $S(\alpha, \beta, \gamma, \delta; \text{param})$ distribution at the value in x.

`SQuantile[p, { $\alpha$ ,  $\beta$ ,  $\gamma$ ,  $\delta$ }, param]` calculates the value of x for a stable $S(\alpha, \beta, \gamma, \delta; \text{param})$ distribution for a quantile p .

`SRandom[n, { $\alpha$ ,  $\beta$ ,  $\gamma$ ,  $\delta$ }, param]` generates n stable $S(\alpha, \beta, \gamma, \delta; \text{param})$ random variables.

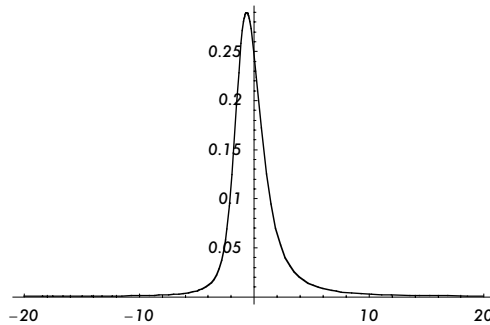
`fMLFit[x_List, param]` fits a $S(\alpha, \beta, \gamma, \delta; \text{param})$ stable distribution to the data in x , using a maximum likelihood method.

`D1SPDF[x, { $\alpha$ ,  $\beta$ ,  $\gamma$ ,  $\delta$ }, param]` calculates the first derivative of the density of a stable $S(\alpha, \beta, \gamma, \delta; \text{param})$ distribution at the value in x .

`SMode[{ $\alpha$ ,  $\beta$ ,  $\gamma$ ,  $\delta$ }, param]` returns the location of the mode of a $S(\alpha, \beta, \gamma, \delta; \text{param})$ stable distribution.

The SPDF function calculates the stable density using a variation of the Zolotarev integral transformation [7]. This transformation permits accurate calculations of both the density and distribution functions across nearly the whole domain of stable parameters. Attempts to simply numerically integrate the inverse Fourier transform of the characteristic function run into difficulties when $\alpha < 1$ or when x is large.

```
In[2]:= Plot[SPDF[x, {1.3, 0.5, 1, 0.5}], {x, -20, 20}];
```



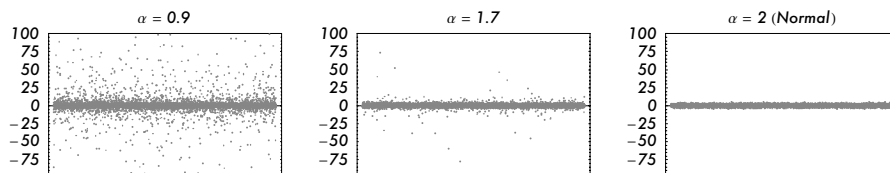
Chambers, Mallows, and Stuck [8] published an algorithm for the generation of stable random variables. It is implemented in this package as the function, `SRandom`. The following example generates a stable random sequence of 10 numbers where $p1$ is a parameter set, $\{\alpha, \beta, \gamma, \delta\}$.

```
In[3]:= p1 = {1.3, 0.5, 1, 0.5};
         SRandom[10, p1]
```

```
Out[4]:= {-0.832182, -0.292768, 0.0465778, 9.15896, -10.9316,
          0.33523, -0.345039, 0.104213, 0.140818, 3.89919}
```

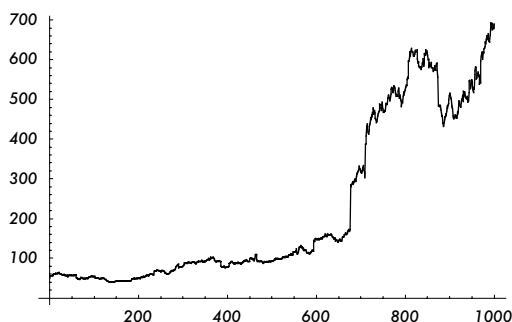
The `SRandom` generator makes it easy to experience the behavior of stable distributions. The following graphics array illustrates 10,000 pseudorandom

drawings from symmetric stable distributions with α set at 0.9, 1.7, and 2.0. Note how extreme outliers may be for lower values of α . The graphs are all at the same scale to illustrate the differences in dispersion, but at this scale some of the extreme points in the case of $\alpha = 0.9$ are not shown. A stable sample with $\alpha = 2$ has the same dispersion as the normal distribution. $\alpha = 1.7$ is typical of stock price returns.



The following example simulates what a series of daily stock market prices might look like if generated by a stable return model with a starting price of 50 and an expected annual return of 7% (there are about 252 trading days in a year); notice the sudden large jumps.

```
In[5]:= ListPlot[50 Exp[FoldList[Plus, 0, SRandom[1000,
    {1.7, -0.1, 0.012, Log[1.07] / 252}]]], PlotJoined -> True];
```



The *StableDistribution* package also has data analysis capability and can perform a fit to data. The `fMLFit` function provides a maximum likelihood fit to the data. The algorithm uses a fast Fourier transform of the characteristic function [9] to approximate the stable probability density function rapidly.

```
In[6]:= p1 = {1.4, 0.9, 1, -0.5}
s4 = SRandom[10000, p1];
p2 = fMLFit[s4]
```

```
Out[6]= {1.4, 0.9, 1, -0.5}
```

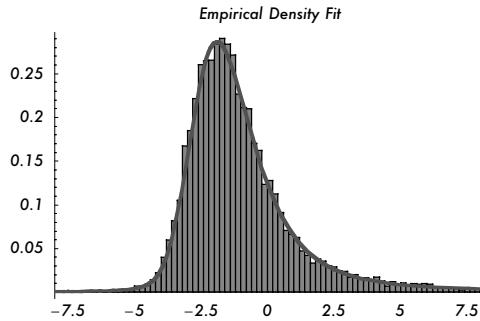
```
FindMaximum::lstol :
```

```
The line search decreased the step size to within tolerance specified by
AccuracyGoal and PrecisionGoal but was unable to find a sufficient
increase in the function. You may need more than MachinePrecision
digits of working precision to meet these tolerances. More . . .
```

```
Out[8]= {1.35908, 0.851204, 0.982561, -0.383498}
```

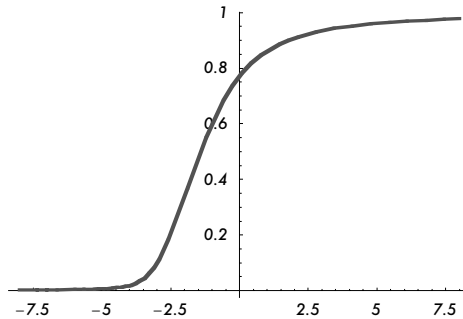
The following graph compares the fit to the empirical density of the sample.

```
In[9]:= << Graphics`Graphics`
empiricdens = Histogram[s4, HistogramScale → 1,
  HistogramRange → {-8, 8}, DisplayFunction → Identity];
calcdens = Plot[SPDF[x, p2], {x, -8, 8}, PlotStyle →
  {Hue[0.67], Thickness[0.008]}, DisplayFunction → Identity];
Show[empiricdens, calcdens, PlotLabel → "Empirical Density Fit",
  DisplayFunction → $DisplayFunction];
```



The plot of the distribution function follows.

```
In[13]:= Plot[SCDF[x, p2], {x, -8, 8},
  PlotStyle → {Hue[0.67], Thickness[0.008]}];
```



The `SQuantile` function uses `FindRoot` to solve for x at a value of the SCDF. It can be used to print a table of cumulative probabilities. For the parameters in the preceding example:

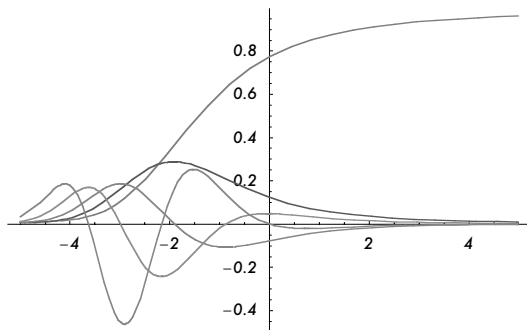
```
In[14]:= Table[{p, SQuantile[p, p1]}, {p, 0, 1, 0.1}] // TableForm
```

```
Out[14]//TableForm=
```

0	$-\infty$
0.1	-3.04994
0.2	-2.55465
0.3	-2.16935
0.4	-1.81086
0.5	-1.44195
0.6	-1.02835
0.7	-0.516218
0.8	0.221794
0.9	1.69672
1.	∞

The density and distribution functions use `NIntegrate` to make the calculations. A direct integral form for the first derivative of the density function is also provided, but further derivatives can also be calculated using the derivative function in *Mathematica*.

```
In[15]:= st1 = Table[Hue[i], {i, 0.6, 1, 0.1}];
f2[x_, a_] := D[SPDF[x, a], {x, 2}];
f3[x_, a_] := D[SPDF[x, a], {x, 3}];
Plot[Evaluate[
  {SCDF[x, p2], SPDF[x, p2], D1SPDF[x, p2], f2[x, p2], f3[x, p2]},
  {x, -5, 5}, PlotStyle -> st1, PlotRange -> All];
```



While the package is able to calculate most all of the range of stable distributions, it can at times be slow and may require adjustment of the input parameters to `NIntegrate` to obtain accurate results. With such adjustments, however, it is possible to make extremely precise calculations. The following example shows high-precision density calculations for values on the light tail of a maximally skewed stable distribution.

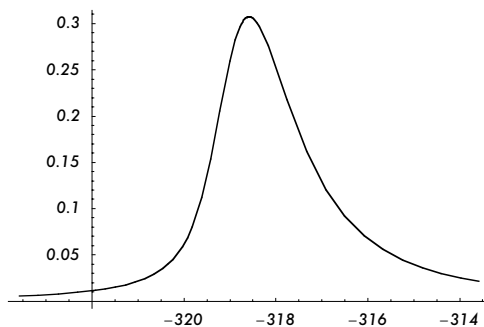
```
In[19]:= Table[{2^i, SPDF[2^i, {13/10, -1}, GaussPoints -> 30,
MaxRecursion -> 10, WorkingPrecision -> 64]}, {i, 6}] // TableForm

Out[19]//TableForm=
2      0.273549462258796189509775341539313131858185752986508966233
4      0.038598879516744249569924583917870716560367065955283947236
8      3.391979139387401921303888873307117869238901361946908063×10-25
16     1.292128069876843683121684007751601752432115597266917711×10-496
32     1.4907823276439596142503265579049423920572129735541813671×10-10006
64     1.28876881056847406907355147746013211220193021391578303×10-201721
```

The `SMode` function can be used to locate the mode of a stable distribution; this is particularly useful when α is close to 1, where the mode may be quite distant from the origin. The function uses `FindRoot` to determine where the first derivative of the stable density is equal to 0.

```
In[20]:= p3 = {1.001, 0.5};
md = SMode[p3]
Plot[SPDF[x, p3], {x, md - 5, md + 5}];
```

```
Out[21]= -318.575
```



■ Program Accuracy, Limitations, and Versatility

The functions illustrated here are designed to have the look and feel of true mathematical functions. `SPDF` uses the Zolotarev transformation of the inverse Fourier transform of the characteristic function in the $S(\alpha, \beta, \gamma, \delta; 0)$ parameterization. The integrand is represented in the program symbolically so that transformation to the $S(\alpha, \beta, \gamma, \delta; 1)$ parameterization is exact. The `SCDF` and `D1SPDF` integrands are created by taking the integral and derivative of the `SPDF` integrand with respect to x before proceeding with transform integration. The function, `NIntegrate`, gives a numerical approximation to the integral. It is important to remember that `NIntegrate` may not always come up with the correct answer. This is particularly true if the integrand has a sharp narrow spike. The integrand for the stable density can be problematic. The integrand is continuous and does not oscillate over the interval of integration; it begins with a value of 0, rises to a peak, $\frac{1}{e}$, and falls to 0 at the other end of the interval. Unfortunately the peak for

some values of x , α , and β can be quite narrow and it is possible for `NIntegrate` to miss the peak with the default settings, resulting in an inaccurate approximation. Since the integrand is otherwise well behaved, it appears to be almost always possible to overcome this limitation and produce calculations to extreme accuracy. When the warning messages for `NIntegrate` are turned on, the first message will generally give a hint as to how to proceed. If the goal is high precision, the first strategy should be to increase the `WorkingPrecision` option. If machine precision is acceptable but the problem is that the peak is missed (most common with probability density function calculations), the best strategy is to increase the `GaussPoints` option. Even though the integrand does not contain a singularity, sometimes it will be necessary to increase the `SingularityDepth` option to catch the peak.

```
In[23]:= OnNIntMsg ; (*Turn on suppressed NIntegrate warning messages*)
```

```
In[24]:= SPDF[0.001, {1.3, 0}]
```

```
NIntegrate::ncvb :  
NIntegrate failed to converge to prescribed accuracy after 7  
recursive bisections in SMath`Private`θ near  
SMath`Private`θ = 0.0006014037137896312`. More . . .
```

```
Out[24]= 0.293983
```

Here we could increase the recursion limit as suggested by the error message, but for these functions it is often a better idea to start with `GaussPoints` which sets a higher number of initial points, thus requiring less recursive subdivision. The difference between the calculations is small.

```
In[25]:= SPDF[0.001, {1.3, 0}, GaussPoints -> 15]  
% - %
```

```
Out[25]= 0.293983
```

```
Out[26]= -4.14103 × 10-10
```

For extreme precision, start with working precision. As the output precision will be limited by the specification of the precision of the input values, it is simplest to use rational numbers. `GaussPoints` and/or `MaxRecursion` will also likely need to be increased. The default setting for `PrecisionGoal` in `NIntegrate` is `WorkingPrecision` minus 10 digits.

```
In[27]:= Table[SPDF[1/1000, {13/10, 0}, GaussPoints -> 30, MaxRecursion -> 10,
WorkingPrecision -> p], {p, 16, 64, 4}] // TableForm
```

```
Out[27]//TableForm=
0.2939835
0.2939834576205198
0.2939834576205198
0.2939834576205197617402
0.29398345762051976174019
0.293983457620519761740188001
0.29398345762051976174018800081232
0.29398345762051976174018800081231973
0.29398345762051976174018800081231973391
0.29398345762051976174018800081231973390709049
0.29398345762051976174018800081231973390709048777
0.293983457620519761740188000812319733907090487774355
0.2939834576205197617401880008123197339070904877743551734
```

When the default `NIntegrate` parameters are used, the program may fail to give accurate results when α is close to 1 or when α is 1 and β is close to 0. These limitations can often be overcome by changing the optional arguments to `NIntegrate`. With the warning messages for `NIntegrate` turned on, *Mathematica* will give notification that requested precision has not been achieved.

```
In[28]:= SPDF[-10, {1.01, 0.5, 1, 0.5}]
```

```
NIntegrate::ploss :
Numerical integration stopping due to loss of precision. Achieved neither
the requested PrecisionGoal nor AccuracyGoal; suspect one of the
following: highly oscillatory integrand or the true value of the integral
is 0. If your integrand is oscillatory on a (semi-)infinite interval
try using the option Method->Oscillatory in NIntegrate. More . . .
```

```
Out[28]= 1.34339 × 10-27
```

```
In[29]:= SPDF[-10, {1.01, 0.5, 1, 0.5}, GaussPoints -> 30]
```

```
Out[29]= 0.00108592
```

```
In[30]:= Plot[SPDF[x, {1.005, 0.02}], {x, -20, 20}, PlotRange -> All];
```

```
NIntegrate::ploss :
Numerical integration stopping due to loss of precision. Achieved neither
the requested PrecisionGoal nor AccuracyGoal; suspect one of the
following: highly oscillatory integrand or the true value of the integral
is 0. If your integrand is oscillatory on a (semi-)infinite interval
try using the option Method->Oscillatory in NIntegrate. More . . .
```

```
NIntegrate::ploss :
Numerical integration stopping due to loss of precision. Achieved neither
the requested PrecisionGoal nor AccuracyGoal; suspect one of the
following: highly oscillatory integrand or the true value of the integral
is 0. If your integrand is oscillatory on a (semi-)infinite interval
try using the option Method->Oscillatory in NIntegrate. More . . .
```

NIntegrate::ploss :

Numerical integration stopping due to loss of precision. Achieved neither the requested PrecisionGoal nor AccuracyGoal; suspect one of the following: highly oscillatory integrand or the true value of the integral is 0. If your integrand is oscillatory on a (semi-)infinite interval try using the option Method->Oscillatory in NIntegrate. More . . .

General::stop : Further output of NIntegrate::ploss will be suppressed during this calculation. More . . .

NIntegrate::ncvb :

NIntegrate failed to converge to prescribed accuracy after 7 recursive bisections in SMath'Private'\theta near SMath'Private'\theta = 1.5092880787986058'. More . . .

NIntegrate::slwcon :

Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration being 0, oscillatory integrand, or insufficient WorkingPrecision. If your integrand is oscillatory try using the option Method->Oscillatory in NIntegrate. More . . .

NIntegrate::ncvb :

NIntegrate failed to converge to prescribed accuracy after 7 recursive bisections in SMath'Private'\theta near SMath'Private'\theta = 1.5092880787986058'. More . . .

NIntegrate::slwcon :

Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration being 0, oscillatory integrand, or insufficient WorkingPrecision. If your integrand is oscillatory try using the option Method->Oscillatory in NIntegrate. More . . .

NIntegrate::ncvb :

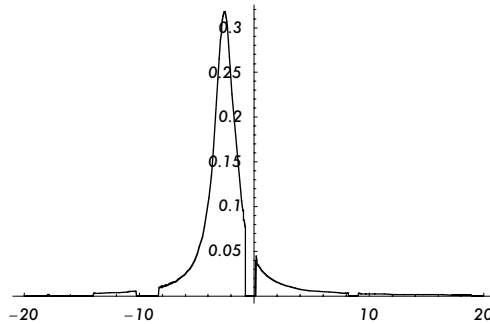
NIntegrate failed to converge to prescribed accuracy after 7 recursive bisections in SMath'Private'\theta near SMath'Private'\theta = 1.5092880787986058'. More . . .

General::stop : Further output of NIntegrate::ncvb will be suppressed during this calculation. More . . .

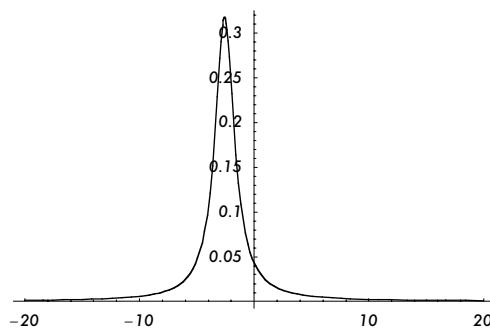
NIntegrate::slwcon :

Numerical integration converging too slowly; suspect one of the following: singularity, value of the integration being 0, oscillatory integrand, or insufficient WorkingPrecision. If your integrand is oscillatory try using the option Method->Oscillatory in NIntegrate. More . . .

General::stop : Further output of NIntegrate::slwcon will be suppressed during this calculation. More . . .



```
In[31]:= Plot[SPDF[x, {1.005, 0.02}], GaussPoints -> 30, MaxRecursion -> 10],
          {x, -20, 20}, PlotRange -> All];
```



The package's Help Browser gives additional examples to help handle error messages generated by the other functions. It contains a section that demonstrates other methods of calculating stable distributions using *Mathematica* and includes the derivation of the Zolotarev integral. There is also an algorithm to set up an interpolation method for approximating the density function for maximum likelihood calculations.

■ **MathLink Implementation**

The direct use of *Mathematica* for data analysis can be slow. For in-depth analysis of data, *STABLE* is an excellent *MathLink* package that includes many more functions and parameterizations. The *MathLink* program consists of compiled Fortran routines that are passed to *Mathematica* with wrapper functions that are compatible with the functions in the *StableDistribution* package. It solves the problem of the shape of the Zolotarev integrand by dividing it into intervals. The peak is found numerically and a narrow interval on either side of the peak is handled with a more involved routine than the other parts. Its compiled code can thus use simpler and faster numerical integration routines for greater speed than possible with *Mathematica*'s very versatile but complicated `NIntegrate` function. It is also bulletproof, at the cost of not allowing calculations when $\text{Abs}[\alpha - 1] < 0.01$ but rounding to $\alpha = 1$, and not permitting calculations when

$\alpha < 0.1$. It is limited to double-precision calculations. The interface can perform maximum likelihood calculations many times faster than the *Mathematica* package described earlier, using a very fast interpolation method. These fits are usually a minimum of 50 times faster than possible with the package described in this paper. The *MathLink* interface has been used with *Mathematica*'s *J/Link* to download stock market data from internet databases and automatically analyze the time series of thousands of stocks. The *MathLink* interface is available (currently for Windows only) from www.robustanalysis.com. The package described in this paper is designed so that both programs can be run simultaneously turning *Mathematica* into a powerful, stable workstation.

■ References

- [1] P. Levy, *Calcul des Probabilites*, Paris: Gauthier-Villars, 1925.
- [2] J. P. Nolan, *Stable Distributions: Models for Heavy Tailed Data*, Boston: Birkhauser, 2005. Chapter 1 provides a good introduction and is available at the American University Stable website academic2.american.edu/~jpnolan/stable/stable.html.
- [3] B. B. Mandelbrot, "The Variation of Certain Speculative Prices," *Journal of Business*, **36**(4), 1963 pp. 394–419.
- [4] B. B. Mandelbrot, "Correction of an Error in "The Variation of Certain Speculative Prices" (1963)," *Journal of Business*, **45**(4), 1972 pp. 542–543.
- [5] B. B. Mandelbrot, *Fractals and Scaling in Finance: Discontinuity, Concentration, Risk*, New York: Springer-Verlag, 1997.
- [6] C. Rose and M. D. Smith, *Mathematical Statistics with Mathematica*, Springer Texts in Statistics, New York: Springer-Verlag, 2002.
- [7] J. P. Nolan, "Numerical Calculation of Stable Densities and Distribution Functions," *Communications in Statistics-Stochastic Models*, **13**(4), 1997 pp. 759–774.
- [8] J. M. Chambers, C. L. Mallows, and B. W. Stuck, "A Method for Simulating Stable Random Variables," *Journal of the American Statistical Association*, **71**(354), 1976 pp. 340–344.
- [9] S. T. Rachev and S. Mittnik, *Stable Paretian Models in Finance*, New York: John Wiley, 2000.
- [10] J. P. Nolan, "Stable MathLink Package," (Mar 18, 2005) www.robustanalysis.com.
- [11] V. M. Zolotarev, "One-Dimensional Stable Distributions," Translation of Mathematical Monographs, Vol. 65, American Mathematical Society, 1986. (Translation of the original 1983 Russian).

■ Additional Material

StableDistribution.zip

Available at www.mathematica-journal.com/issue/v9i4/download.

About the Authors

Robert H. Rimmer received his M.D. degree from the State University of New York. He is a clinical cardiologist with an interest in mathematics. He became interested in stable distributions after reading *The Fractal Geometry of Nature* by B. B. Mandelbrot in the 1980s. He has been programming with *Mathematica* as a hobby since 1988.

John P. Nolan received his Ph.D. in mathematics from the University of Virginia. He is a professor at American University in Washington, DC, where he teaches and does research on stable distributions and statistical genetics.

Robert H. Rimmer

HC 4, Box 3-C
Payson, AZ 85541-9571
rrimmer@npgcable.com

John P. Nolan

Math/Stat Department
227 Gray Hall
American University
4400 Massachusetts Avenue, NW
Washington, DC 20016-8050
jpnolan@american.edu

Computational Order Statistics

Colin Rose
Murray D. Smith

Using **mathStatICA** [1], Rose and Smith [2] illustrate the automated derivation of the exact density of order statistics obtained from a random sample of size n on a *continuous* random variable. This article illustrates the new functionality in **mathStatICA** to allow for *discrete* random variables. Moreover, by building on the new **Piecewise** capability in *Mathematica*, we are able to further generalise to the case of non-identically distributed random variables, thereby providing a completely flexible solution.

■ Introduction

Let X denote a continuous random variable with probability density function (pdf) $f(x)$ and cumulative distribution function (cdf) $F(x)$, and let $(X_1, X_2, \dots, X_n)$ denote a random sample of size n drawn on X . Let $(X_{(1)}, X_{(2)}, \dots, X_{(n)})$ denote the random sample ordered such that $X_{(1)} < X_{(2)} < \dots < X_{(n)}$; then $(X_{(1)}, X_{(2)}, \dots, X_{(n)})$ are collectively known as the *order statistics derived from the parent X* . For example, $X_{(1)} = \min(X_1, \dots, X_n)$ is the smallest order statistic and corresponds to the sample minimum, and $X_{(n)}$ is the largest order statistic and corresponds to the sample maximum.

For a detailed discussion of the statistical theory pertaining to order statistics see, for example, [3, 4, 5, 6]. For the case in which X is a discrete random variable, see [7]. On the computational side, Rose and Smith [2, Section 9.4] use **mathStatICA** to obtain the algebraic and numeric properties of order statistics derived from a continuous parent. Whereas Evans et al. [8] appear to be restricted to numeric-only calculations, they also consider settings in which the parent variable is sampled without replacement.

Section 1 illustrates briefly the case of a continuous parent, while Sections 2 and 3 extend to the case of a discrete parent. Section 4 relaxes the independent and identically distributed (iid) assumptions, thereby illustrating some of the new functionality of **mathStatICA**.

We begin by loading **mathStatICA** 1.5 or later.

```
In[1]:= << mathStatICA.m
```

■ 1. Continuous Parent Distribution

Let the continuous random variable X have a U-shaped distribution with pdf $f(x)$:

$$\text{In}[2]:= \quad f = \frac{1}{\pi \sqrt{A^2 - x^2}};$$

defined on a domain of support $(-A, A)$, where parameter $A > 0$:

$$\text{In}[3]:= \quad \text{domain}[f] = \{x, -A, A\} \&\& \{A > 0\};$$

Figure 1 plots the parent pdf's when parameter $A = 1, 3$ and 5 :

$$\text{In}[4]:= \quad \text{PlotDensity}[f /. A \rightarrow \{1, 3, 5\}];$$

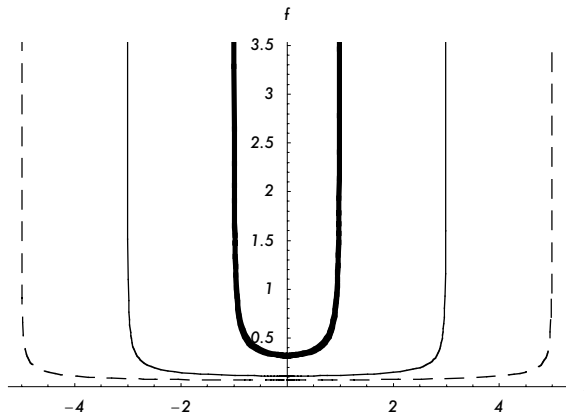


Figure 1.

For a sample of size n on X , the pdf of the r^{th} order statistic $X_{(r)}$ is given by the `mathStat` function:

$$\text{In}[5]:= \quad g = \text{OrderStat}[r, f]$$

$$\text{Out}[5]= \frac{2^{1-n} \pi^{-n} \left(\pi - 2 \text{ArcTan} \left[\frac{x}{\sqrt{A^2 - x^2}} \right] \right)^{n-r} \left(\pi + 2 \text{ArcTan} \left[\frac{x}{\sqrt{A^2 - x^2}} \right] \right)^{-1+r} n!}{\sqrt{(A-x)(A+x)} (n-r)! (-1+r)!}$$

with domain of support:

$$\text{In}[6]:= \quad \text{domain}[g] = \text{OrderStatDomain}[r, f]$$

$$\text{Out}[6]= \quad \{x, -A, A\} \&\& \{n \in \text{Integers}, r \in \text{Integers}, A > 0, 1 \leq r \leq n\}$$

Figure 2 plots the pdf of $X_{(r)}$ as r increases from 1 to 10, given a sample size $n = 10$, with $A = 3$:

```
In[7]:= PlotDensity[g /. {A → 3, r → Range[10], n → 10}];
```

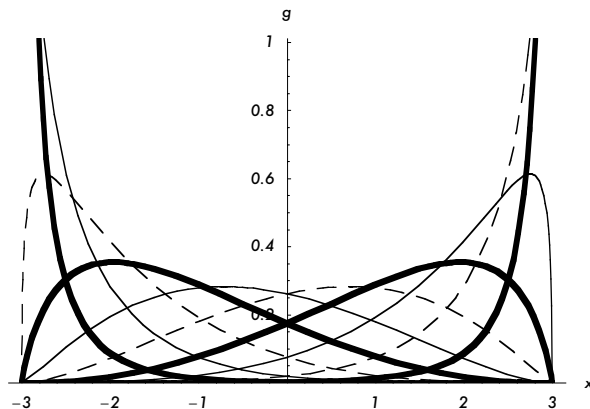


Figure 2.

The bivariate pdf of two order statistics $X_{(r)}$ and $X_{(s)}$, for $r < s$, is given by:

```
In[8]:= OrderStat[{r, s}, f]
```

$$\text{Out[8]} = \left(\pi^{-1+r-s} \left(\frac{1}{2} + \frac{\text{ArcTan}\left[\frac{x_r}{\sqrt{A^2 - x_r^2}}\right]}{\pi} \right)^{-1+r} \right. \\ \left. \left(-\text{ArcTan}\left[\frac{x_r}{\sqrt{A^2 - x_r^2}}\right] + \text{ArcTan}\left[\frac{x_s}{\sqrt{A^2 - x_s^2}}\right] \right)^{-1-r+s} \right. \\ \left. \left(\frac{1}{2} - \frac{\text{ArcTan}\left[\frac{x_s}{\sqrt{A^2 - x_s^2}}\right]}{\pi} \right)^{n-s} \Gamma[1+n] \right) / \\ \left(\Gamma[r] \Gamma[1+n-s] \Gamma[-r+s] \sqrt{(A^2 - x_r^2)(A^2 - x_s^2)} \right)$$

■ 2. Discrete Parent Distribution (Function Form)

mathStatICA 1.5 expands its functionality to the case of a discrete parent, again for *arbitrary* distributions. To illustrate, let the random variable X have a Poisson two-component-mix distribution with probability mass function (pmf) $f(x)$:

$$\text{In[9]}:= f = \omega \frac{e^{-\lambda} \lambda^x}{x!} + (1-\omega) \frac{e^{-\theta} \theta^x}{x!};$$

and domain of support:

```
In[10]:= domain[f] = {x, 0, ∞} && {\lambda > 0, \theta > 0, 0 < \omega < 1} && {Discrete};
```

Here, θ and λ are the Poisson parameters, while ω is the mixing weight parameter. Figure 3 illustrates this pmf when $\lambda = 4$, $\theta = 20$ and $\omega = \frac{1}{2}$:

```
In[11]:= PlotDensity[f /. {λ → 4, θ → 20, ω → 1/2}];
```

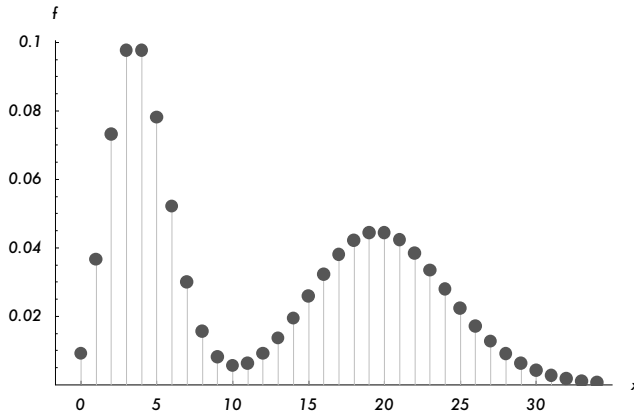


Figure 3.

Given a sample of size n , the pmf of the r^{th} order statistic $X_{(r)}$, denoted $g(x)$, is:

```
In[12]:= g = OrderStat[r, f]
```

```
Out[12]= 1 / Beta[r, 1 + n - r] *
  (-Beta[ x * (-(-1 + ω) Gamma[x, θ] + ω Gamma[x, λ]) / Γ[1 + x], r, 1 + n - r] +
   Beta[ -(1 + x) * ((-1 + ω) Gamma[1 + x, θ] - ω Gamma[1 + x, λ]) / Γ[2 + x], r, 1 + n - r])
```

with domain of support:

```
In[13]:= domain[g] = OrderStatDomain[r, f]
```

```
Out[13]= {x, 0, ∞} && {n ∈ Integers, r ∈ Integers,
  θ > 0, λ > 0, 0 < ω < 1, 1 ≤ r ≤ n} && {Discrete}
```

Figure 4 plots the pmf of the minimum order statistic (i.e. $r = 1$) when $\lambda = 4$, $\theta = 20$ and $\omega = \frac{1}{2}$ and the sample size is $n = 10$:

```
In[14]:= param = {λ → 4, θ → 20, ω →  $\frac{1}{2}$ , r → 1, n → 10};
PlotDensity[g /. param];
```

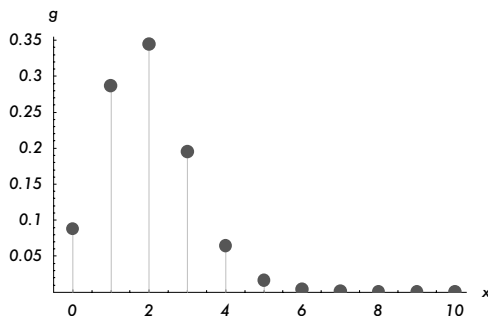


Figure 4.

□ Monte Carlo 'Check' of the Exact Solution We Have Just Plotted

Here is a single pseudorandom sample of size $n = 10$ drawn from the parent two-component-mix Poisson pmf $f(x)$:

```
In[16]:= DiscreteRNG[10, f /. param]
```

```
Out[16]= {3, 3, 22, 32, 12, 15, 2, 19, 4, 3}
```

If we want 50,000 such samples (each of size 10), the neatest approach is to generate all $50000 * 10$ drawings in one go:

```
In[17]:= data = DiscreteRNG[50000 * 10, f /. param]; // Timing
```

```
Out[17]= {1.96 Second, Null}
```

and then partition this data into 50,000 samples (each of size 10). We can then find the minimum of each of the 50,000 samples by mapping the Min function across each sample:

```
In[18]:= samplemin = Map[Min, Partition[data, 10]];
```

If n is *very* large, efficient algorithms specifically designed for pseudorandom generation of order statistics exist; see [9]. Figure 5 plots the *empirical* relative frequency distribution of the sample minimum data ($\blacktriangle$) together with the *exact* pmf of the sample minimum ($\bullet$): a good match should see the former obscure the latter nearly everywhere.

```
In[19]:= FrequencyPlotDiscrete[samplemin, g /. param];
```

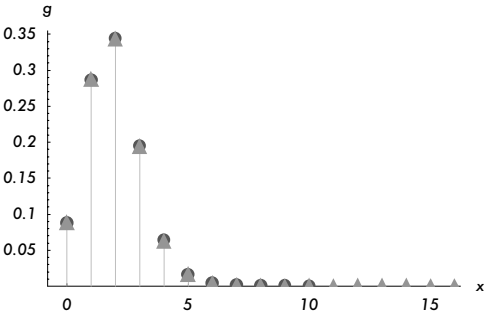


Figure 5.

■ 3. Discrete Parent Distribution (List Form)

Suppose we throw an *unfair* six-sided die onto a flat surface such as a table. Let X denote the upmost face of the die with pmf:

$P(X = x) :$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{12}$	$\frac{3}{12}$
$x :$	1	2	3	4	5	6

Table 1. The pmf of X .

Using List Form, we enter this pmf as follows:

```
In[20]:= f = {1/6, 1/6, 1/6, 1/6, 1/12, 3/12};
domain[f] = {x, {1, 2, 3, 4, 5, 6}} && {Discrete};
```

The pmf of the r^{th} order statistic $X_{(r)}$ is:

```
In[22]:= g = OrderStat[r, f]
```

Out[22]=
$$\left\{ \frac{\text{Beta}[\frac{1}{6}, r, 1+n-r]}{\text{Beta}[r, 1+n-r]}, \frac{-\text{Beta}[\frac{1}{6}, r, 1+n-r] + \text{Beta}[\frac{1}{3}, r, 1+n-r]}{\text{Beta}[r, 1+n-r]}, \right.$$
$$\frac{-\text{Beta}[\frac{1}{3}, r, 1+n-r] + \text{Beta}[\frac{1}{2}, r, 1+n-r]}{\text{Beta}[r, 1+n-r]},$$
$$\frac{-\text{Beta}[\frac{1}{2}, r, 1+n-r] + \text{Beta}[\frac{2}{3}, r, 1+n-r]}{\text{Beta}[r, 1+n-r]},$$
$$\frac{-\text{Beta}[\frac{2}{3}, r, 1+n-r] + \text{Beta}[\frac{3}{4}, r, 1+n-r]}{\text{Beta}[r, 1+n-r]},$$
$$\left. \frac{-\text{Beta}[\frac{3}{4}, r, 1+n-r] + \text{Beta}[1, r, 1+n-r]}{\text{Beta}[r, 1+n-r]} \right\}$$

with domain of support:

```
In[23]:= domain[g] = OrderStatDomain[r, f]
```

```
Out[23]= {x, {1, 2, 3, 4, 5, 6}} &&
          {n ∈ Integers, r ∈ Integers, 1 ≤ r ≤ n} && {Discrete}
```

Here, for example, is a plot of the pmf of the third order statistic $X_{(3)}$ when the sample size $n = 10$:

```
In[24]:= PlotDensity[g /. {r → 3, n → 10}];
```

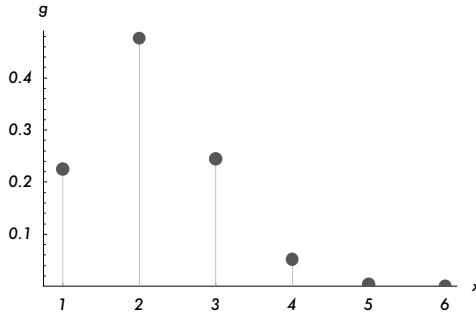


Figure 6.

■ 4. Extensions and Forthcoming Features

Thus far, this article has assumed that we are dealing with samples of iid variables. In this section, we take the major step of relaxing these assumptions. The generalisation to non-identical distributions is an enormously flexible and powerful capability. To do so, we require the new `Piecewise` functionality found in *Mathematica* 5.1 or later. In the examples that follow, we provide an illustration/preview of this new functionality as already implemented in the developmental version of *mathStat*ica and which will be available in its next public release.

□ Non-Identical Parameters

Let X_i denote a continuous random variable with pdf $f(x; \lambda_i)$ and cdf $F(x; \lambda_i)$, such that $(X_1, X_2, \dots, X_n)$ are independent but *not identical* variables due to differing parameters λ_i , for $i = 1, \dots, n$. For example, consider an $\text{Exponential}(\lambda)$ parent where indentity is relaxed by replacing λ with λ_i , for $i = 1, \dots, n$. Thus:

```
In[25]:= f = 1/λi e-x/λi ; domain[f] = {x, 0, ∞} && {λi > 0};
```

Then, the pdf of the minimum order statistic (in, say, a sample of size 4) is:

In[26]:= **OrderStat[1, f₁, 4]**

$$\text{Out[26]} = \frac{e^{-x \left(\frac{1}{\lambda_1} + \frac{1}{\lambda_2} + \frac{1}{\lambda_3} + \frac{1}{\lambda_4} \right)} (\lambda_2 \lambda_3 \lambda_4 + \lambda_1 (\lambda_3 \lambda_4 + \lambda_2 (\lambda_3 + \lambda_4)))}{\lambda_1 \lambda_2 \lambda_3 \lambda_4}$$

The pdf of the next largest order statistic, $X_{(2)}$, is substantially more complicated:

In[27]:= **OrderStat[2, f₁, 4]**

$$\begin{aligned} \text{Out[27]} = & \frac{1}{\lambda_1 \lambda_2 \lambda_3 \lambda_4} \left(e^{-x \left(\frac{1}{\lambda_1} + \frac{1}{\lambda_2} + \frac{1}{\lambda_3} + \frac{1}{\lambda_4} \right)} \right. \\ & \left((-3 + e^{\frac{x}{\lambda_2}} + e^{\frac{x}{\lambda_3}} + e^{\frac{x}{\lambda_4}}) \lambda_2 \lambda_3 \lambda_4 + \lambda_1 \left((-3 + e^{\frac{x}{\lambda_1}} + e^{\frac{x}{\lambda_3}} + e^{\frac{x}{\lambda_4}}) \lambda_3 \lambda_4 + \right. \right. \\ & \left. \left. \lambda_2 \left((-3 + e^{\frac{x}{\lambda_1}} + e^{\frac{x}{\lambda_2}} + e^{\frac{x}{\lambda_3}}) \lambda_3 + (-3 + e^{\frac{x}{\lambda_1}} + e^{\frac{x}{\lambda_2}} + e^{\frac{x}{\lambda_4}}) \lambda_4 \right) \right) \right) \end{aligned}$$

□ Non-Identical Distributions

Next, let us suppose we have three completely different distributions defined over three different domains of support. In the following, $f(x)$ is the pdf of an Exponential(λ), $g(x)$ is the pdf of a standard Normal, and $h(x)$ is the pdf of a Uniform($-1, 1$) random variable:

$$\text{In[28]} := f = \frac{1}{\lambda} e^{-x/\lambda}; \quad \text{domain}[f] = \{x, 0, \infty\} \&\& \{\lambda > 0\};$$

$$\text{In[29]} := g = \frac{e^{-\frac{x^2}{2}}}{\sqrt{2\pi}}; \quad \text{domain}[g] = \{x, -\infty, \infty\};$$

$$\text{In[30]} := h = \frac{1}{2}; \quad \text{domain}[h] = \{x, -1, 1\};$$

We can now solve completely general questions. For example, let us suppose we have a random sample of size $n = 20$. Of this sample, suppose that 10 values are drawn from the Normal, seven from the Exponential, and three from the Uniform. What is the pdf of the second smallest value from the sample, namely the second order statistic? Solving this problem would normally be enormously complicated, but the solution is now given simply by:

In[31]:= **OrderStat[2, {f, g, h}, {10, 7, 3}]**

The output can be viewed in the electronic version of this notebook.

This same technology provides a neat way to solve problems such as finding the pdf of $\min(X, Y, Z)$, when X, Y and Z have completely different distributions and different domains of support. For our example, if $X \sim \text{Exponential}(\lambda)$, $Y \sim \text{Normal}(0, 1)$ and $Z \sim \text{Uniform}(-1, 1)$, then the pdf of $\min(X, Y, Z)$ is simply the pdf of the first order statistic:

```
In[32]:= sol = OrderStat[1, {f, g, h}]
```

$$\text{Out[32]} = \begin{cases} \frac{e^{-\frac{x^2}{2}}}{\sqrt{2\pi}} & x \leq -1 \\ -\frac{e^{-\frac{x^2}{2}}(-1+x)}{2\sqrt{2\pi}} + \frac{1}{4} \operatorname{Erfc}\left[\frac{x}{\sqrt{2}}\right] & -1 < x \leq 0 \\ \frac{e^{-\frac{1}{2}x\left(x+\frac{2}{\lambda}\right)}\left(-\sqrt{\frac{2}{\pi}}(-1+x)\lambda + e^{\frac{x^2}{2}(1-x+\lambda)}\operatorname{Erfc}\left[\frac{x}{\sqrt{2}}\right]\right)}{4\lambda} & 0 < x < 1 \end{cases}$$

with domain of support:

```
In[33]:= domain[sol] = {x, -∞, ∞} && {λ > 0};
```

Here is a plot of the pdf we have just derived:

```
In[34]:= PlotDensity[sol /. λ → 1, {x, -4, 2}, PlotStyle → Hue[1]];
```

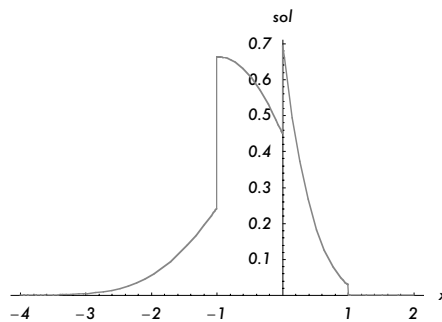


Figure 7.

We can easily ‘check’ our solution using Monte Carlo methods. Here are 100,000 pseudorandom drawings from each of the three distributions:

```
In[35]:= << Statistics`
```

```
In[36]:= dataf = RandomArray[ExponentialDistribution[1/λ] /. λ → 1, 100000];
          datag = RandomArray[NormalDistribution[0, 1], 100000];
          datah = RandomArray[UniformDistribution[-1, 1], 100000];
```

Next, we create 100,000 samples of size 3 containing one drawing from each of the three distributions, and then map the `Min` function across each sample, generating our 100,000 empirical drawings of the sample minimum:

```
In[39]:= samplemin = Map[Min, Transpose[{dataf, datag, datah}]];
```

Figure 8 compares the empirical pdf (—) of the data we have just generated with the theoretical pdf (---) derived earlier:

```
In[40]:= FrequencyPlot[samplemin, {-4, 2, .05}, sol /. λ → 1];
```

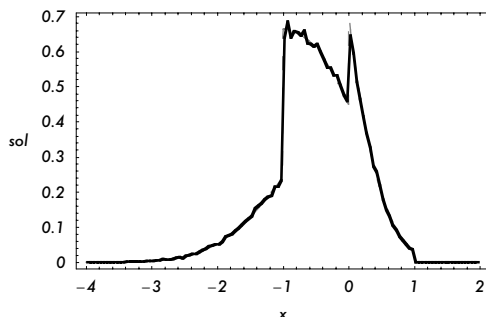


Figure 8.

□ Independence Relaxed, Identity Maintained

The distribution of an order statistic is derived as a many-to-one transformation from the joint distribution of $(X_1, \dots, X_n)$. Although there are differing ways in which the distribution can be found, perhaps the simplest method uses equivalence events. To illustrate, the event $X_{(n)} \leq x$ is equivalent to $X_i \leq x$, for all $i = 1, \dots, n$, where x is arbitrarily chosen. Accordingly, the cdf of $X_{(n)}$ in terms of x is given by

$$P(X_{(n)} \leq x) = P(X_1 \leq x, \dots, X_n \leq x). \quad (1)$$

In the case of $X_{(n)}$ (and $X_{(1)}$ too), there is only one equivalent event. The number of equivalent events increases substantially as other inner order statistics are considered; however, in order to keep our discussion as simple as possible we will confine attention to $X_{(n)}$ from here on. This is not necessarily a case without interest, for the extremal $X_{(n)}$ is encountered in many practical contexts. Examples include $X_{(n)}$ as the measure of record high temperatures and record times in sports.

If the standard iid assumptions hold, then $(X_1, \dots, X_n)$ is a collection of mutually independent random variables, and all are copies of the same parent X (with pdf $f(x)$ and cdf $F(x)$), leading to considerable simplification in the right-hand side of (1):

$$P(X_{(n)} \leq x) = P(X \leq x)^n. \quad (2)$$

If, for example, X is continuous, then the pdf of $X_{(n)}$ is obtained by differentiation of (2) with respect to x , yielding $g(x) = n F(x)^{n-1} f(x)$.

Just as identity can be relaxed in many ways, so too can independence. To introduce a dependence structure, we may begin by rewriting (1) in its copula form

$$\begin{aligned} P(X_{(n)} \leq x) &= C(F_1(x), \dots, F_n(x)) \\ &= C(F(x), \dots, F(x)), \end{aligned} \quad (3)$$

where the second line recognises that $X_1, \dots, X_n$ are copies of a common parent X . The n -copula $C: [0, 1]^n \rightarrow [0, 1]$ is the function that represents the dependence structure of $X_1, \dots, X_n$. For example, the special case $C(u_1, \dots, u_n) = u_1 \dots u_n$ corresponds to mutual independence amongst $X_1, \dots, X_n$. The representation (3) is due to Sklar [10] and is unique provided X is continuous.

Tractable results can be obtained by assuming an Archimedean dependence structure for C ; for details of these copulas see [11, Chapter 4]. Let $\varphi: [0, 1] \rightarrow [0, \infty]$ denote the strict generator function associated with C ; it is differentiable such that $\varphi'(t) = \partial \varphi(t) / \partial t < 0$ on $0 < t < 1$, and its inverse φ^{-1} must be completely monotonic if $n \geq 3$. For example, the generator associated with the independence copula $\varphi(t) = -\log t$ is strict, and its inverse $\varphi^{-1}(t) = \exp(-t)$ is completely monotonic. Then, the key property of the generator is that

$$\begin{aligned}\varphi(G(x)) &= \varphi(F(x)) + \dots + \varphi(F(x)) \\ &= n \varphi(F(x)),\end{aligned}\tag{4}$$

where $G(x) = P(X_{(n)} \leq x)$ is the cdf of $X_{(n)}$. Differentiating both sides of (4) with respect to x and rearranging yields the pdf of $X_{(n)}$:

$$g(x) = n \frac{\varphi'(F(x))}{\varphi'(G(x))} f(x),\tag{5}$$

where the denominator would be computed as per $\varphi'(G(x)) = \varphi'(\varphi^{-1}(n \varphi(F(x))))$. The resemblance in the structure of the pdf (5) to the pdf in the iid case, $n F(x)^{n-1} f(x)$, is striking.

To illustrate, let $X \sim \text{Exponential}(\lambda)$ with pdf $f(x)$:

```
In[41]:= f = 1/λ e-x/λ ; domain[f] = {x, 0, ∞} && {λ > 0};
```

and cdf $F(x) = P(X \leq x)$:

```
In[42]:= F = Prob[x, f]
```

```
Out[42]= 1 - e-x/λ
```

and summarise our assumptions:

```
In[43]:= assum = {n ∈ Integers, n > 0, x > 0, λ > 0, θ ≥ 1};
```

Enter the details for a particular case considered by Ballerini [12], namely, that of the Gumbel-Hougaard family of n -copulas with generator $\varphi(t) = (-\log t)^\theta$, with dependence parameter $\theta \geq 1$:

```
In[44]:= φ[t_] = (-Log[t])θ; φi[t_] = Exp[-t1/θ]; φd[t_] = D[φ[t], t];
```

Then, from (5), the pdf of $X_{(n)}$ is given by

```
In[45]:= g = FullSimplify[n  $\frac{\varphi d[F]}{\varphi d[\varphi i[n * \varphi[F]]}$  f, assum]
```

$$\text{Out[45]} = \frac{(1 - e^{-x})^{\frac{1}{\theta}} n^{\frac{1}{\theta}}}{(-1 + e^{x/\lambda}) \lambda}$$

with domain of support:

```
In[46]:= domain[g] = {x, 0, ∞} && assum;
```

Setting $\theta = 1$ corresponds to the iid case; notice too that replacing n in the iid pdf with $n^{1/\theta}$ yields the pdf of $X_{(n)}$. This, for example, means that many algebraic results on the properties of $X_{(n)}$ can be found simply by replacing n with $n^{1/\theta}$ in the appropriate iid formula. In Figure 9, the solid line denotes the pdf when $\theta = 1$ (the iid case), while the dashed line denotes the pdf when $\theta = 2$.

```
In[47]:= PlotDensity[g /. {θ → {1, 2}, λ → 1, n → 10}];
```

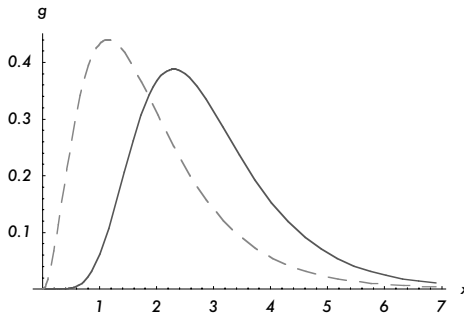


Figure 9.

■ References

- [1] mathStatICA, Sydney, Australia: MATHSTATICA Pty. Ltd., (Apr 2005) www.mathStatICA.com.
- [2] C. Rose and M. D. Smith, *Mathematical Statistics with Mathematica*, Springer Texts in Statistics, New York: Springer-Verlag, 2002.
- [3] H. A. David and H. N. Nagaraja, *Order Statistics*, 3rd ed., New York: John Wiley & Sons, 2003.
- [4] N. Balakrishnan and C. R. Rao, eds., *Order Statistics: Theory and Methods*, Handbook of Statistics, Vol. 16, Amsterdam: Elsevier, 1998.
- [5] ———, *Order Statistics: Applications*, Handbook of Statistics, Vol. 17, Amsterdam: Elsevier, 1998.
- [6] B. C. Arnold, N. Balakrishnan, and H. N. Nagaraja, *A First Course in Order Statistics*, New York: John Wiley & Sons, 1992.
- [7] H. N. Nagaraja, "Order Statistics from Discrete Distributions," *Statistics*, **23**, 1992 pp. 189–216.

- [8] D. L. Evans, L. M. Leemis, and J. H. Drew, "The Distribution of Order Statistics for Discrete Random Variables with Applications for Bootstrapping," *INFORMS Journal on Computing (JOC)*, forthcoming, 2005.
- [9] P. R. Tadikamalla and N. Balakrishnan, "Computer Simulation of Order Statistics," in Balakrishnan and Rao, pp. 65–72.
- [10] A. Sklar, "Random Variables, Joint Distributions, and Copulas," *Kybernetika*, **9**, 1973 pp. 449–460.
- [11] R. B. Nelsen, *An Introduction to Copulas*, Lecture Notes in Statistics, New York: Springer-Verlag, 1999.
- [12] R. Ballerini, "A Dependent F^α -Scheme," *Statistics & Probability Letters*, **21**(1), 1994 pp. 21–25.

About the Authors

Colin Rose is director of the Theoretical Research Institute, Sydney. In 1998, he was a visiting scholar at Wolfram Research. His recent research interests relate to exact computational methods in mathematical statistics, and in particular, with application to the mathStatistica project.

Murray D. Smith is an associate professor in the Discipline of Econometrics and Business Statistics at the University of Sydney. His recent research includes applications of computational algebra in statistics and the uses of copulas in modelling dependence structures in econometrics.

Colin Rose

*Theoretical Research Institute
Sydney NSW 2023
Australia
colin@tri.org.au*

Murray D. Smith

*Econometrics and Business Statistics
The University of Sydney
Sydney NSW 2006
Australia
Murray.Smith@econ.usyd.edu.au*

Applications of Generating Functions in Nonparametric Tests

Peter Weiß

The combination of *Mathematica*'s high algebraic capacity and method of generating functions is becoming an extremely efficient tool in probability theory and statistics. After an introductory example and a short overview on nonparametric methods, we show how generating functions of discrete statistics can be handled using *Mathematica*. Next, we solve two combinatorial problems, which are essential in calculating the generating function of nonparametric test statistics. Finally, these methods are used to produce accurate and extensive tables and plots of the distributions of some of the most popular nonparametric test statistics. Furthermore, we use graphs to illustrate approximations of the distributions of these test statistics by standard distributions.

■ An Introductory Example

We start our investigation with an example, which is used in the *Statistica* [1] manual to explain the Kruskal–Wallis analysis of variance with tied observations.

$n = 15$ small children who were randomly assigned to one of $k = 3$ experimental groups (treatments) of equal size, $n_1 = n_2 = n_3 = 5$. Each child was shown a series of pairs of stimuli. Their task was to choose one of those stimuli, and, if it was the “correct” one, they received a reward. The relevant dimensions that the children had to detect to make correct choices were form (group 1), color (group 2), and size (group 3). The dependent variable was the number of trials the children required to detect the dimension that was being rewarded. The following table shows the data set.

Group 1	9	10	11	8	9
Group 2	10	8	15	10	11
Group 3	12	10	12	17	15

We have to test if the distributions of the groups' performances (the numbers of trials required to detect the relevant dimensions) differ significantly from each

other. Because there is no reason to assume that the performances are normally distributed, we can not apply the usual analysis of variance. Following [2], we have to use the Kruskal–Wallis analysis of variance with tied observations (the values 8, 9, 10, 11, 12, and 15 occur several times and are called ties).

Therefore, we determine the different performances, their frequencies, and their midranks

Performance	Frequency	Midrank	Performance	Frequency	Midrank
8	2	1.5	12	2	11.5
9	2	3.5	15	2	13.5
10	4	6.5	17	1	15
11	2	9.5			

and calculate the sum ($R_1 = 24.5$, $R_2 = 37.5$, $R_3 = 58$) of the midranks. We have to reject the null hypothesis (the performances of the three groups are equally distributed) and accept the alternative hypothesis (the performances of these three groups are not equally distributed) if the Kruskal–Wallis statistic with tied observations

$$K = \frac{1}{\Delta} \left[\frac{12}{n(n+1)} \sum_{i=1}^k \frac{1}{n_i} R_i^2 - 3(n+1) \right]$$

with

$$\Delta = 1 - \frac{1}{n^3 - n} \sum_{j=1}^l (d_j^3 - d_j)$$

is greater than or equal to a certain value, $\kappa_{1-\alpha}$. In this formula, d_j denotes the frequency of the j -th performance. $\kappa_{1-\alpha}$ (the so-called $1-\alpha$ quantile of the Kruskal–Wallis statistic with tied observations) is the greatest real number x with

$$\Pr(\{K < x\}) < 1 - \alpha.$$

The aim of this article is to show how to calculate the distribution and especially the quantiles of such nonparametric test statistics using *Mathematica*. Applying these methods to this introductory example, we find $\kappa_{0.95} = 5.65651$. Thus, because of

$$K = 5.85062 > 5.65651 = \kappa_{0.95},$$

we conclude that the distributions of the performances of the different groups differ significantly (with a level of significance of $\alpha = 5\%$) from each other.

■ Nonparametric Tests

Nonparametric methods were developed to be used when the researcher knows nothing about the distribution of the variable of interest (hence the name *nonpara-*

metric). Basically, there is at least one nonparametric equivalent for each parametric type of test. When you want to compare the distribution of two independent samples, usually you would use the Student t -test; one of the most important alternatives for this test is the Wilcoxon rank sum test. If you have more than two independent samples to compare, usually you would use analysis of variance; the nonparametric equivalent to this method is the Kruskal–Wallis analysis of variance.

Nonparametric tests are robust (they need no assumptions about the distribution of the background population), efficient (in early days it was believed that a heavy price in loss of efficiency would have to be paid for robustness, but [2, 3] and several other authors showed clearly that the efficiency is comparable with classical tests using the assumption of normality), and easy to handle using computers.

Nevertheless, they are not widely accepted. One possible reason is that the application of a nonparametric test requires the use of (and the confidence in) a table (see, for example, [4] or [5]) of the distribution of its test statistics.

Generating such tables requires heavy algebraic manipulations and is therefore mostly beyond the scope of introductory textbooks. Published tables are sometimes inaccurate or not extensive enough (especially in the case of ties). Moreover, Mitic [6] pointed out that the entries of some published tables differ, depending on their source. Finally, if the sample size is large, it is possible to approximate most of these distributions by standard distributions, but little is known about the quality of these approximations for small sample sizes.

Thus, I believe, *Mathematica* users are interested in procedures that allow them to generate accurate and extensive tables of the distributions of nonparametric test statistics and plot the distribution functions of these statistics. With these procedures, *Mathematica* users are independent of published tables and can investigate the quality of the approximation by standard distributions.

■ Generating Functions

Roughly speaking, a generating function is a polynomial in one or more variables (in expanded form), whose exponents are real numbers and whose coefficients are the numbers we are seeking. Generating functions are widely used in probability theory ([7], [8], or [9]) and combinatorics [10]. In this article we use the word “polynomial” in a nonstandard sense. Usually polynomials have integer exponents only. Because *Mathematica* works well with this kind of “generalized polynomials,” we use them instead.

For instance, if *Stat* is a discrete statistic with possible values $x_1, x_2, \dots, x_n$, its generating function is defined as

$$G[\text{Stat}] = \sum_{i=1}^n \Pr(\{\text{Stat} = x_i\}) t^{x_i},$$

where the argument t plays only an auxiliary role. Thus, if $Stat$ is a binomial distributed with parameters n and p , we have

$$G[Stat] = \sum_{i=0}^n \Pr(\{Stat = i\}) t^i = \sum_{i=0}^n \binom{n}{i} p^i (1-p)^{n-i} t^i = (pt + (1-p))^n.$$

The generating function $G[Stat]$ of a discrete statistic $Stat$ contains all the information about its distribution. Given the generating function, it is thus possible to calculate (as well as plot or tabulate)

- the probability density function, $PDFG[Stat, x] := \Pr(\{Stat = x\})$;
- the cumulated distribution function, $CDFG[Stat, x] := \Pr(\{Stat < x\})$;
- the alpha-quantile, $QTLG[Stat, \alpha] := \max\{x \mid DFG[Stat, x] < \alpha\}$;
- the n -th moment, $MOMG[Stat, n]$; and
- the variance, $VARG[Stat]$

of $Stat$ (the letter “G” in the name of these functions indicates that we base our calculations on the generating function $G[Stat]$ of $Stat$). The following easy and relatively fast procedures are useful in this context:

```
In[25]:= PDFG[Stat_, x_] := Coefficient[G[Stat], t, x];
```

```
In[26]:= CDFG[Stat_, x_] :=
Module[{glist, g, ex, exinf, coef, cum, i = 0},
  glist = Apply[List, G[Stat]];
  g = Length[glist];
  ex = Table[Exponent[glist[[i]], t], {i, 1, g}];
  exinf = Append[ex, Infinity];
  coef = glist /. t -> 1;
  cum = FoldList[Plus, 0, coef];
  Scan[If[#1 >= x, Return[cum[[i + 1]]], i++] &, exinf];
```

```
In[27]:= PlotCDFG[Stat_, opt___] :=
Module[{glist, g, ex, exmod, coef, cum, tbl},
  glist = Apply[List, G[Stat]];
  g = Length[glist];
  ex = Table[Exponent[glist[[i]], t], {i, 1, g}];
  exmod = Prepend[Append[ex, Last[ex] + 1], First[ex] - 1];
  coef = glist /. t -> 1;
  cum = FoldList[Plus, 0, coef];
  tbl = Table[{exmod[[i + j]], cum[[j]]}, {j, 1, g + 1}, {i, 0, 1}];
  Show[Graphics[Line[Flatten[tbl, 1]]], opt];
```

```

In[28]:= TableCDFG[Stat_, xmin_, xmax_] :=
Module[{glist, g, ex, exinf, coef, cum, imin, imax, i = 0, j = 0},
  glist = Apply[List, G[Stat]];
  g = Length[glist];
  ex = Table[Exponent[glist[[i]], t], {i, 1, g}];
  exinf = Append[ex, Infinity];
  coef = glist /. t -> 1;
  cum = FoldList[Plus, 0, coef];
  imin = Scan[If[#1 >= xmin, Return[i + 1], i++] &, ex];
  imax = Scan[If[#1 >= xmax, Return[j], j++] &, exinf];
  TableForm[Table[N[{ex[[i]], cum[[i]]}], {i, imin, imax}],
    TableHeadings -> {None, {"x", "CDF[x]"}},
    TableSpacing -> {0.5, 5}];

In[29]:= QTLG[Stat_, α_] :=
Module[{glist, g, ex, coef, cum, i = 0},
  glist = Apply[List, G[Stat]];
  g = Length[glist];
  ex = Table[Exponent[glist[[i]], t], {i, 1, g}];
  coef = glist /. t -> 1;
  cum = FoldList[Plus, 0, coef];
  Scan[If[#1 >= α, Return[ex[[i]]], i++] &, cum];

In[30]:= pol[Stat_, 1] := D[G[Stat], t];
pol[Stat_, n_] := D[pol[Stat, n - 1] t, t];
MOMG[Stat_, n_] := pol[Stat, n] /. t -> 1;
VARG[Stat_] := MOMG[Stat, 2] - MOMG[Stat, 1]^2;

```

Now we give an example of an application of generating functions in combinatorics. For instance, if we divide n numbered balls randomly among k urns and denote by $A[\{n_1, n_2, \dots, n_k\}]$ the number of cases that n_1 balls come into urn $U_1, \dots, n_k$ balls come into urn U_k , then the generating function of these numbers is defined as

$$\begin{aligned}
 a[n, k] &= \sum_{\substack{n_1, \dots, n_k=1 \\ n_1 + \dots + n_k = n}}^n A[\{n_1, n_2, \dots, n_k\}] x[1]^{n_1} \dots x[k]^{n_k} = \\
 &= \sum_{\substack{n_1, \dots, n_k=1 \\ n_1 + \dots + n_k = n}}^n \frac{n!}{n_1! \dots n_k!} x[1]^{n_1} \dots x[k]^{n_k} = \left(\sum_{i=1}^k x[i] \right)^n,
 \end{aligned}$$

where the arguments $x[i]$ again play only an auxiliary role.

■ Two Combinatorial Problems

The crucial point in nonparametric test theory is that all possible arrangements of the ranks of the observed values are equally likely. To calculate, for example, the distribution density $\Pr(\{Stat = x\})$ of a statistic $Stat$, which is based on ranks, it is therefore only necessary to obtain the number of cases fulfilling the condi-

tion $Stat = x$. The two following combinatorial problems give essential ideas in doing this.

□ Problem 1

Suppose we have n balls, which are numbered $1, 2, \dots, n$. In how many different ways

$$P[\{n_1, \dots, n_k\}, \{r_1, \dots, r_k\}]$$

is it possible to divide these n balls among k urns, such that

- the i -th urn U_i contains n_i balls and
- the sum of the numbers of these n_i balls in urn U_i is r_i

with $n = n_1 + \dots + n_k$ and $r = r_1 + \dots + r_k = \frac{1}{2} n(n+1)$?

For small n and k we can calculate this number by counting the relevant partitions. For example, there are the following 15 partitions of $n = 6$ balls into $k = 2$ urns with $n_1 = 4$ balls in urn U_1 and $n_2 = 2$ balls in urn U_2 :

U_1	r_1	U_2	r_2	U_1	r_1	U_2	r_2	U_1	r_1	U_2	r_2
{1, 2, 3, 4}	10	{5, 6}	11	{1, 2, 3, 6}	12	{4, 5}	9	{1, 3, 5, 6}	15	{2, 4}	6
{1, 2, 3, 5}	11	{4, 6}	10	{1, 2, 4, 6}	13	{3, 5}	8	{2, 3, 5, 6}	16	{1, 4}	5
{1, 2, 4, 5}	12	{3, 6}	9	{1, 3, 4, 6}	14	{2, 5}	7	{1, 4, 5, 6}	16	{2, 3}	5
{1, 3, 4, 5}	13	{2, 6}	8	{2, 3, 4, 6}	15	{1, 5}	6	{2, 4, 5, 6}	17	{1, 3}	4
{2, 3, 4, 5}	14	{1, 6}	7	{1, 2, 5, 6}	14	{3, 4}	7	{3, 4, 5, 6}	18	{1, 2}	3

Thus, $P[\{4, 2\}, \{12, 9\}] = 2$ and $P[\{4, 2\}, \{14, 7\}] = 3$. If n and k are not as small as in this example, this method fails because there are too many partitions of n balls into k urns with n_i balls in urn U_i .

□ Problem 2

Suppose we have n balls, which are marked by real numbers in the following way: d_1 of these balls have mark m_1 , d_2 have mark m_2 , $\dots$, and d_l have mark m_l (with $n = d_1 + \dots + d_l$). In how many different ways

$$Q[\{d_1, \dots, d_l\}, \{m_1, \dots, m_l\}, \{n_1, \dots, n_k\}, \{r_1, \dots, r_k\}]$$

is it possible to divide these n balls among k urns, such that

- the i -th urn U_i contains n_i balls and
- the sum of the marks of these n_i balls is r_i

with $n = n_1 + \dots + n_k = d_1 + \dots + d_l$ and $r = r_1 + \dots + r_k = d_1 m_1 + \dots + d_l m_l$?

This problem is obviously a generalization of our first problem, since for $l = n$ we have

$$Q[\{1, \dots, 1\}, \{1, \dots, n\}, \{n_1, \dots, n_k\}, \{r_1, \dots, r_k\}] = P[\{n_1, \dots, n_k\}, \{r_1, \dots, r_k\}].$$

For small n and k it is again easy to calculate $Q[\text{dlist}, \text{mlist}, \text{nlist}, \text{rlist}]$ by counting the relevant partitions. If we mark $d_1 = 3$ of the $n = 6$ balls with the mark $m_1 = 1$ (let us assume that these are the balls with the numbers 1, 2, 3) and the remaining $d_2 = 3$ balls with the mark $m_2 = 2$, and if we put again $n_1 = 4$ of these balls in urn U_1 and $n_2 = 2$ of these balls in urn U_2 , we obtain (the same) 15 different partitions of our marked balls. (In the following table we show only their marks.)

U_1	r_1	U_2	r_2	U_1	r_1	U_2	r_2	U_1	r_1	U_2	r_2
{1, 1, 1, 2}	5	{2, 2}	4	{1, 1, 1, 2}	5	{2, 2}	4	{1, 1, 2, 2}	6	{1, 2}	3
{1, 1, 1, 2}	5	{2, 2}	4	{1, 1, 2, 2}	6	{1, 2}	3	{1, 1, 2, 2}	6	{1, 2}	3
{1, 1, 2, 2}	6	{1, 2}	3	{1, 1, 2, 2}	6	{1, 2}	3	{1, 2, 2, 2}	7	{1, 1}	2
{1, 1, 2, 2}	6	{1, 2}	3	{1, 1, 2, 2}	6	{1, 2}	3	{1, 2, 2, 2}	7	{1, 1}	2
{1, 1, 2, 2}	6	{1, 2}	3	{1, 1, 2, 2}	6	{1, 2}	3	{1, 2, 2, 2}	7	{1, 1}	2

Thus $Q[\{3, 3\}, \{1, 2\}, \{4, 2\}, \{6, 3\}] = 9$ and $Q[\{3, 3\}, \{1, 2\}, \{4, 2\}, \{7, 2\}] = 3$. If n and k are not as small as in this example, this method fails again.

□ A Simple Method

The aim of this section is to find the generating functions for the numbers $P[\text{nlist}, \text{rlist}]$ and $Q[\text{dlist}, \text{mlist}, \text{nlist}, \text{rlist}]$ with $\text{nlist} := \{n_1, \dots, n_k\}$, $\text{rlist} := \{r_1, \dots, r_k\}$, $\text{dlist} := \{d_1, \dots, d_l\}$, and $\text{mlist} := \{m_1, \dots, m_l\}$.

With some experience it is easy to see that our generating functions must have two sorts of variables: for each urn U_i one variable $x[i]$ governing the number of balls in this urn and one variable $y[i]$ governing the sum of the numbers (marks) of the balls in this urn. With this remark in mind, the generating functions for the numbers $P[\text{nlist}, \text{rlist}]$ and $Q[\text{dlist}, \text{mlist}, \text{nlist}, \text{rlist}]$ are obviously the polynomials

$$p[n, k] = \prod_{j=1}^n \left(\sum_{i=1}^k x[i] y[i]^j \right),$$

$$q[\text{dlist}, \text{mlist}, k] = \prod_{j=1}^l \left(\sum_{i=1}^k x[i] y[i]^{m_j} \right)^{d_j}.$$

To be more precise, the numbers $P[nlist, rlist]$ and $Q[dlist, mlist, nlist, rlist]$ are the coefficients of

$$\prod_{i=1}^k x[i]^{n_i} y[i]^{r_i}$$

of the polynomials $p[n, k]$ and $q[dlist, mlist, k]$.

In *Mathematica*, we define these two polynomials $p[n, k]$ and $q[dlist, mlist, k]$ recursively:

```
In[34]:= xlist[k_] := Table[x[i], {i, 1, k}];
        ylist[k_] := Table[y[i], {i, 1, k}];

In[36]:= p[1, k_] := xlist[k].ylist[k];
        p[n_, k_] := p[n, k] = p[n-1, k] xlist[k].ylist[k]^n

In[38]:= q[{d_}, {m_}, k_] := (xlist[k].ylist[k]^m)^d;
        q[dlist_, mlist_, k_] := q[dlist, mlist, k] =
            q[{First[dlist]}, {First[mlist]}, k] q[Rest[dlist], Rest[mlist], k]

In[40]:= Expand[p[3, 2]]

Out[40]= x[1]^3 y[1]^6 + x[1]^2 x[2] y[1]^5 y[2] + x[1]^2 x[2] y[1]^4 y[2]^2 +
        x[1]^2 x[2] y[1]^3 y[2]^3 + x[1] x[2]^2 y[1]^3 y[2]^3 +
        x[1] x[2]^2 y[1]^2 y[2]^4 + x[1] x[2]^2 y[1] y[2]^5 + x[2]^3 y[2]^6

In[41]:= Expand[q[{2, 2}, {1, 2}, 2]]

Out[41]= x[1]^4 y[1]^6 + 2 x[1]^3 x[2] y[1]^5 y[2] +
        2 x[1]^3 x[2] y[1]^4 y[2]^2 + x[1]^2 x[2]^2 y[1]^4 y[2]^2 +
        4 x[1]^2 x[2]^2 y[1]^3 y[2]^3 + x[1]^2 x[2]^2 y[1]^2 y[2]^4 +
        2 x[1] x[2]^3 y[1]^2 y[2]^4 + 2 x[1] x[2]^3 y[1] y[2]^5 + x[2]^4 y[2]^6
```

Now we get the numbers $P[nlist, rlist]$ and $Q[dlist, mlist, nlist, rlist]$ by selecting the coefficients of the polynomials $p[n, k]$ and $q[dlist, mlist, k]$:

```
P[nlist_, rlist_] :=
Module[{n, r, k},
  n = Apply[Plus, nlist];
  r = Apply[Plus, rlist];
  k = Length[nlist];
  If[r == n (n+1) / 2,
    Fold[Coefficient, Expand[p[n, k]],
      Union[xlist[k]^nlist, ylist[k]^rlist]],
    Print["r ≠ n(n+1)/2"]]]
```

```

Q[dlist_, mlist_, nlist_, rlist_] :=
Module[{d, n, r, k},
  d = Apply[Plus, dlist];
  n = Apply[Plus, nlist];
  r = Apply[Plus, rlist];
  k = Length[nlist];
  If[n == d && r == dlist.mlist,
    Fold[Coefficient, Expand[q[dlist, mlist, k]],
      Union[xlist[k]^nlist, ylist[k]^rlist]],
    Print["n ≠ d1+...+d1 or r ≠ d1 m1+...+d1 m1"]]]

```

Using these functions, we solve the combinatorial problems posed at the beginning of this section and a problem, which can not be solved by hand.

```
In[44]:= P[{4, 2}, {14, 7}]
```

```
Out[44]= 3
```

```
In[45]:= Q[{3, 3}, {1, 2}, {4, 2}, {6, 3}]
```

```
Out[45]= 9
```

```

Timing[
Q[{3, 4, 5, 4}, {1, 5, 2, 3}, {5, 5, 4, 2}, {15, 12, 11, 7}]]
{24.81 Second, 109440}

```

□ A More Sophisticated Method

Unfortunately this easy method wastes time and memory (for instance, the last calculation needs more than 100 MB RAM). This disadvantage is because the number of terms of the polynomials $p[n, k]$ and $q[dlist, mlist, k]$ are of order k^n , which is a huge number even if n and k are not very large. We overcome this difficulty by fixing $nlist = \{n_1, \dots, n_k\}$. In this case the number of terms of the generating functions $p[nlist]$ and $q[dlist, mlist, nlist]$ of $P[nlist, rlist]$ and $Q[dlist, mlist, nlist, rlist]$ are only of order $\text{Multinomial}[n_1, \dots, n_k]$. This is also a large number, but an essentially smaller one than k^n .

It is obvious that these new generating functions $p[nlist]$ and $q[dlist, mlist, nlist]$ are the coefficients of

$$\prod_{i=1}^k x[i]^{n_i}$$

of our old polynomials $p[n, k]$ and $q[dlist, mlist, k]$. Of course we do not use this fact, since it would not save any memory. What we do is calculate these new generating functions $p[nlist]$ and $q[dlist, mlist, nlist]$ recursively using the obvious formulas

$$p[\{n_1, \dots, n_k\}] = \sum_{i=1}^k y[i]^n p[\{n_1, \dots, n_i - 1, \dots, n_k\}]$$

$$q[\{d_1, \dots, d_l\}, \{m_1, \dots, m_l\}, \{n_1, \dots, n_k\}] =$$

$$\sum_{i=1}^k y[i]^{m_i} q[\{d_1, \dots, d_i - 1\}, \{m_1, \dots, m_l\}, \{n_1, \dots, n_i - 1, \dots, n_k\}]$$

and some self-evident boundary conditions, if some of the n_i 's are zero or d_l is zero.

```

In[47]:= p[{n1_}] := y[1]^(n1 (n1 + 1) / 2);
p[nlist_] := p[nlist] =
Module[{n, k, id, nlistnew, pos, sub},
  n = Apply[Plus, nlist];
  k = Length[nlist];
  id = IdentityMatrix[k];
  nlistnew = Delete[nlist, Position[nlist, 0]];
  pos = Flatten[Position[nlist, j_ /; j > 0]];
  sub = Table[y[i] -> y[pos[[i]]], {i, 1, Length[pos]}];
  If[FreeQ[nlist, 0],
    Expand[Sum[y[i]^n p[nlist - id[[i]]], {i, 1, k}]],
    p[nlistnew] /. sub]]

q[{d_}, {m_}, nlist_] := Apply[Multinomial, nlist]
  Apply[Times, ylist[Length[nlist]]^(m nlist)];
q[dlist_, mlist_, nlist_] := q[dlist, mlist, nlist] =
Module[{d, n, k, id, dlistnew, nlistnew, pos, sub},
  d = Apply[Plus, dlist];
  n = Apply[Plus, nlist];
  k = Length[nlist];
  id = IdentityMatrix[k];
  dlistnew = Append[Drop[dlist, -1], Last[dlist] - 1];
  nlistnew = Delete[nlist, Position[nlist, 0]];
  pos = Flatten[Position[nlist, j_ /; j > 0]];
  sub = Table[y[i] -> y[pos[[i]]], {i, 1, Length[pos]}];
  Which[d != n, Print["n != d1+...+dl"]; Abort[],
    ! (FreeQ[nlist, 0]), q[dlist, mlist, nlistnew] /. sub,
    ! (FreeQ[dlist, 0]), q[Drop[dlist, -1], Drop[mlist, -1], nlist],
    FreeQ[nlist, 0] && FreeQ[dlist, 0],
    Expand[Sum[y[i]^Last[mlist]
      q[dlistnew, mlist, nlist - id[[i]]], {i, 1, k}]]]]

In[51]:= p[{4, 2}]

Out[51]= y[1]^18 y[2]^3 + y[1]^17 y[2]^4 + 2 y[1]^16 y[2]^5 + 2 y[1]^15 y[2]^6 + 3 y[1]^14 y[2]^7 +
  2 y[1]^13 y[2]^8 + 2 y[1]^12 y[2]^9 + y[1]^11 y[2]^10 + y[1]^10 y[2]^11

```

```
In[52]:= q[{2, 1, 1}, {1, 3, 2}, {1, 2, 1}]
```

```
Out[52]= 2 y[1]^3 y[2]^3 y[3] + 2 y[1]^2 y[2]^4 y[3] +
          2 y[1] y[2]^5 y[3] + y[1]^3 y[2]^2 y[3]^2 +
          2 y[1] y[2]^4 y[3]^2 + y[1]^2 y[2]^2 y[3]^3 + 2 y[1] y[2]^3 y[3]^3
```

Now we get $P[nlist, rlist]$ and $Q[dlist, mlist, nlist, rlist]$ by selecting the coefficients of

$$\prod_{i=1}^k y[i]^{r_i}$$

of the polynomials $p[nlist]$ and $q[dlist, mlist, nlist]$:

```
P[nlist_, rlist_] :=
Module[{n, r, k},
  n = Apply[Plus, nlist];
  r = Apply[Plus, rlist];
  k = Length[nlist];
  If[r == n (n + 1) / 2,
    Fold[Coefficient, p[nlist], ylist[k]^rlist],
    Print["r ≠ n(n+1)/2"]]]

Q[dlist_, mlist_, nlist_, rlist_] :=
Module[{d, n, r, k},
  d = Apply[Plus, dlist];
  n = Apply[Plus, nlist];
  r = Apply[Plus, rlist];
  k = Length[nlist];
  If[d == n && r == dlist.mlist,
    Fold[Coefficient, q[dlist, mlist, nlist], ylist[k]^rlist],
    Print["n ≠ d1+...+d1 or r ≠ d1 m1+...+d1 m1"]]]
```

The following examples show the efficiency of this method:

```
In[55]:= P[{3, 4, 5}, {15, 25, 38}]
```

```
Out[55]= 91
```

```
In[56]:= Q[{2, 6, 4}, {1, 2, 4}, {3, 4, 5}, {6, 10, 14}]
```

```
Out[56]= 440
```

```
Timing[
  Q[{3, 4, 5, 4}, {1, 5, 2, 3}, {5, 5, 4, 2}, {15, 12, 11, 7}]]
{3.5 Second, 109440}
```

■ The Wilcoxon Rank Sum Test

Suppose two independent samples $X_1, \dots, X_{n_1}$ and $Y_1, \dots, Y_{n_2}$ of sizes n_1 and n_2 are drawn from two continuous (not necessarily normal) populations. We wish to test the null hypothesis H_0 (these two populations are identically distributed)

against the alternative hypothesis H_1 (these two populations are not identically distributed).

The Wilcoxon rank sum test (which is closely related to the Mann–Whitney U test) is one of the best known and easiest to use tests in this situation (see, for example, [2]). It rejects the null hypothesis H_0 if the sum of the ranks of the X_i 's in the combined ordered arrangement of the two samples is either too large or too small. This statistic is called Wilcoxon rank sum, which we denote by $\text{WilcoxonRS}[n_1, n_2]$.

□ No Ties

Let us first assume that there are no ties, that is, all $n = n_1 + n_2$ observations are different from each other. Calculating the generating function $G[\text{WilcoxonRS}[n_1, n_2]]$ of the null distribution of the Wilcoxon rank sum, it is important to mention that for all

$$0 \leq k_1 \leq \frac{1}{2} n(n+1) \text{ and } k_2 = \frac{1}{2} n(n+1) - k_1,$$

we have

$$\Pr(\{\text{WilcoxonRS}[n_1, n_2] = k_1\}) = \frac{P[\{n_1, n_2\}, \{k_1, k_2\}]}{\text{Binomial}[n, n_1]}.$$

Thus we get

```
In[58]:= G[WilcoxonRS[n1_, n2_]] :=
  Apart[(p[{n1, n2}] /. {y[1] -> t, y[2] -> 1}) / Binomial[n1 + n2, n1]]

In[59]:= G[WilcoxonRS[4, 3]]

Out[59]=  $\frac{t^{10}}{35} + \frac{t^{11}}{35} + \frac{2t^{12}}{35} + \frac{3t^{13}}{35} + \frac{4t^{14}}{35} + \frac{4t^{15}}{35} +$ 
 $\frac{t^{16}}{7} + \frac{4t^{17}}{35} + \frac{4t^{18}}{35} + \frac{3t^{19}}{35} + \frac{2t^{20}}{35} + \frac{t^{21}}{35} + \frac{t^{22}}{35}$ 
```

Together with our procedures of the third section, we can use this generating function to calculate the probability density function; to calculate, tabulate, and plot the cumulated distribution function; and to calculate alpha-quantiles of the null distribution of the Wilcoxon rank sum. For example,

```
In[60]:= CDFG[WilcoxonRS[4, 3], 13]

Out[60]=  $\frac{4}{35}$ 

In[61]:= QTLG[WilcoxonRS[5, 5], 0.90]

Out[61]= 34
```

For large values of n_1 and n_2 , it is well known (see, for example, [2]) that the null distribution of the test statistic $\text{WilcoxonRS}[n_1, n_2]$ can be approximated by an appropriate normal distribution. We can now investigate the quality of this

approximation in the case of relatively small values of n_1 and n_2 graphically (Figure 1).

```
In[62]:= Needs["Statistics`NormalDistribution`"]
In[63]:= n1 = 6; n2 = 3;
mu = MOMG[WilcoxonRS[n1, n2], 1];
sigma = Sqrt[VARG[WilcoxonRS[n1, n2]]];
plot1 = PlotCDFG[WilcoxonRS[n1, n2], DisplayFunction -> Identity];
plot2 = Plot[CDF[NormalDistribution[mu, sigma], x],
  {x, mu - Floor[3 sigma], mu + Ceiling[3 sigma]},
  DisplayFunction -> Identity];
fig1 = Show[{plot1, plot2}, PlotRange -> All,
  DisplayFunction -> $DisplayFunction, Axes -> True]
```

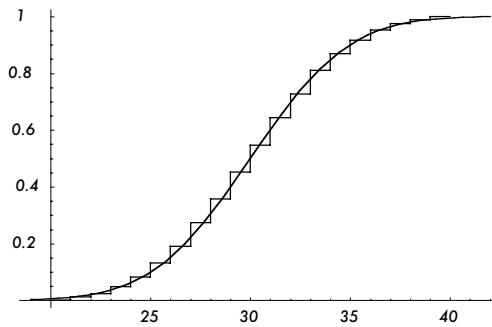


Figure 1.

Furthermore, it is possible to generate accurate tables of alpha-quantiles of the null distribution of this test statistic.

```
In[69]:= n1max = 12; n2max = 12; alpha = 0.90;
TableForm[
  Table[QTLG[WilcoxonRS[i, j], alpha], {i, 2, n1max}, {j, 1, n2max}],
  TableHeadings -> {Prepend[Table[i, {i, 3, n1max}], "n1=2"],
    Prepend[Table[j, {j, 2, n2max}], "n2=1"]},
  TableAlignments -> Center,
  TableSpacing -> {0.5, 1}]
```

Out[70]//TableForm=

	n2=1	2	3	4	5	6	7	8	9	10	11	12
n1=2	5	7	8	10	11	13	15	16	18	19	21	22
3	9	11	13	16	18	20	22	24	27	29	31	33
4	14	17	20	22	25	28	31	34	36	39	42	45
5	20	23	27	30	34	37	41	44	47	51	54	57
6	27	31	35	39	43	47	51	55	59	63	67	71
7	35	40	44	49	54	58	63	67	72	76	81	85
8	44	49	54	60	65	70	75	80	85	91	96	101
9	53	60	66	71	77	83	89	94	100	106	112	117
10	64	71	78	84	91	97	103	110	116	122	128	135
11	76	84	91	98	105	112	119	126	133	139	146	153
12	89	97	105	113	120	128	135	143	150	158	165	172

□ Ties

If ties occur, we assign to tied observations the same midrank, which is the average of the ranks of these observations. Suppose we have the seven observations 1.3, 1.7, 1.7, 1.7, 1.9, 1.9, and 2.2, then the midranks of these observations are 1, 3, 3, 3, 5.5, 5.5, and 7.

If d_1 observations have midrank m_1 and ... d_l observations have midrank m_l and if these midranks are ordered increasingly, we can calculate these midranks using the formula

$$m_j = \sum_{s=1}^{j-1} d_s + \frac{d_j + 1}{2}.$$

Thus, we define

```
In[71]:= m[dlist_] := FoldList[Plus, 0, Drop[dlist, -1]] + (dlist + 1) / 2
```

Calculating the generating function $G[\text{WilcoxonRSTies}[\text{dlist}, n_1, n_2]]$ of the null distribution of the Wilcoxon rank sum (which is now the sum of the midranks of the X_i 's in the combined ordered arrangement of the two samples), it is important to mention that for all

$$0 \leq k_1 \leq \frac{1}{2} n(n+1) \text{ and } k_2 = \frac{1}{2} n(n+1) - k_1$$

we have

$$\Pr(\{\text{WilcoxonRSTies}[\text{dlist}, n_1, n_2] = k_1\}) = \frac{Q[\text{dlist}, \{n_1, n_2\}, \{k_1, k_2\}]}{\text{Binomial}[n, n_1]}.$$

Thus, we get

```
G[WilcoxonRSTies[dlist_, n1_, n2_]] :=
  If[Apply[Plus, dlist] == n1 + n2,
    Apart[(q[dlist, m[dlist], {n1, n2}] /. {y[1] -> t, y[2] -> 1}) /
      Binomial[n1 + n2, n1]],
    Print["n1+n2 ≠ d1+...+dl"]; Abort[]]
```

```
In[73]:= G[WilcoxonRSTies[{2, 3, 4}, 5, 4]]
```

$$\text{Out[73]} = \frac{t^{15}}{126} + \frac{2 t^{37/2}}{21} + \frac{4 t^{21}}{63} + \frac{t^{22}}{7} + \frac{2 t^{49/2}}{7} + \frac{2 t^{51/2}}{63} + \frac{t^{27}}{21} + \frac{4 t^{28}}{21} + \frac{2 t^{61/2}}{21} + \frac{t^{63/2}}{63} + \frac{t^{34}}{42}$$

Again we can use this generating function to calculate the probability density function; to calculate, tabulate, and plot the cumulated distribution function; and to calculate alpha-quantiles of the null distribution of the Wilcoxon rank sum. For example,

```
In[74]:= TableCDFG[WilcoxonRSTies[{3, 3, 4}, 5, 5], 30, 40]
```

```
Out[74]//TableForm=
```

x	CDF [x]
32.	0.761905
32.5	0.785714
35.5	0.928571
36.	0.97619
39.	0.988095

In [2] it is shown that for large values of n_1 and n_2 the null distribution of the test statistics $\text{WilcoxonRSTies}[\text{dlist}, n_1, n_2]$ can be approximated by an appropriate normal distribution. Again we investigate the quality of this approximation in the case of relatively small values of n_1 and n_2 graphically (Figure 2).

```
In[75]:= dlist = {1, 2, 3, 2, 1}; n1 = 5; n2 = 4;
mu = MOMG[WilcoxonRSTies[dlist, n1, n2], 1];
sigma = Sqrt[VARG[WilcoxonRSTies[dlist, n1, n2]]];
plot1 = PlotCDFG[
  WilcoxonRSTies[dlist, n1, n2], DisplayFunction -> Identity];
plot2 = Plot[CDF[NormalDistribution[mu, sigma], x],
  {x, mu - Floor[3 sigma], mu + Ceiling[3 sigma]},
  DisplayFunction -> Identity];
fig2 = Show[{plot1, plot2}, PlotRange -> All,
  DisplayFunction -> $DisplayFunction, Axes -> True]
```

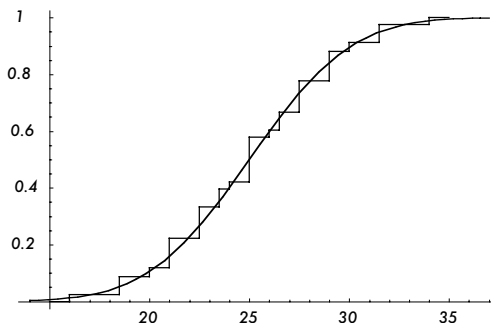


Figure 2.

■ The Kruskal–Wallis Analysis of Variance

The Kruskal–Wallis analysis of variance is a generalization of the Wilcoxon rank sum test. These two tests are related in the same way as the well-known analysis of variance and the two-sample Student t -test.

Suppose that k independent samples $X_{11}, \dots, X_{1n_1}; \dots; X_{k1}, \dots, X_{kn_k}$ of sizes $n_1, \dots, n_k$ are drawn from k continuous (not necessarily normal) populations. We want to test the null hypothesis H_0 (these populations are identically distributed) against the alternative hypothesis H_1 (these populations are not identically distributed).

□ No Ties

Let us assume first that there are no ties. In that case, we calculate the sum $R_1, \dots, R_k$ of the ranks of the $X_{1\bullet}$'s, $\dots$, $X_{k\bullet}$'s in the combined ordered arrangement of these k samples and reject the null hypotheses H_0 , if the Kruskal–Wallis statistics

$$\text{KruskalWallis}[\text{nlist}] = \frac{12}{n(n+1)} \sum_{i=1}^k \frac{1}{n_i} R_i^2 - 3(n+1)$$

with $n = n_1 + \dots + n_k$ is too large.

Calculating the generating function $G[\text{KruskalWallis}[\text{nlist}]]$ of the null distribution of the Kruskal–Wallis test statistics, it is important to mention that

$$\Pr(\{R_1 = r_1\} \cap \dots \cap \{R_k = r_k\}) = \frac{P[\text{nlist}, \text{rlist}]}{\text{Multinomial}[n_1, \dots, n_k]}.$$

Thus, we get the generating function $G[\text{KruskalWallis}[\text{nlist}]]$ by applying the substitution

$$\text{s1} = \left\{ y[i]^r \rightarrow t^{\wedge} \left(\frac{r^2}{n_i} \right) \mid i = 1, \dots, k \right\}$$

to

$$\frac{p[\text{nlist}]}{\text{Multinomial}[n_1, \dots, n_k]}$$

and the substitution

$$\text{s2} = t^x \rightarrow t^{\wedge} \left(\frac{12x}{n(n+1)} - 3(n+1) \right)$$

to the result of the first substitution.

```
In[81]:= G[KruskalWallis[nlist_]] :=
Module[{n, k, a, b, sub1, sub2},
  n = Apply[Plus, nlist];
  k = Length[nlist];
  a = 12 / (n (n + 1));
  b = 3 (n + 1);
  s1 = Table[y[i]^r_ -> t^(r^2/nlist[[i]]), {i, 1, k}];
  s2 = t^x_ -> t^(a x - b);
  Apart[(p[nlist] /. s1 /. s2) / Apply[Multinomial, nlist]]]
```

```
In[82]:= G[KruskalWallis[{2, 3, 2}]]
```

$$\begin{aligned} \text{Out[82]} = & \frac{1}{35} + \frac{8t^{5/28}}{105} + \frac{4t^{3/14}}{105} + \frac{2\sqrt{t}}{35} + \frac{2t^{17/28}}{35} + \frac{2t^{5/7}}{35} + \\ & \frac{t^{6/7}}{35} + \frac{4t^{33/28}}{105} + \frac{2t^{19/14}}{35} + \frac{4t^{41/28}}{105} + \frac{2t^{45/28}}{35} + \frac{t^{27/14}}{35} + \frac{2t^2}{105} + \\ & \frac{4t^{31/14}}{105} + \frac{2t^{17/7}}{105} + \frac{4t^{69/28}}{105} + \frac{4t^{11/4}}{105} + \frac{2t^{20/7}}{105} + \frac{2t^{89/28}}{105} + \frac{t^{24/7}}{105} + \\ & \frac{2t^{101/28}}{105} + \frac{4t^{15/4}}{105} + \frac{8t^{55/14}}{105} + \frac{4t^{125/28}}{105} + \frac{2t^{9/2}}{105} + \frac{2t^{33/7}}{105} + \frac{t^{75/14}}{35} \end{aligned}$$

Therefore, we have, for example,

```
In[83]:= TableCDFG[KruskalWallis[{2, 3, 3}], 4.3, 6.5]
```

```
Out[83]//TableForm=
```

x	CDF[x]
4.55556	0.9
4.69444	0.907143
5.	0.925
5.13889	0.939286
5.36111	0.967857
5.55556	0.975
6.25	0.989286

It is well known (we refer again to Lehmann [2]) that for large values of $n_1, \dots, n_k$ it is possible to approximate the null distribution of the test statistics $\text{KruskalWallis}[nlist]$ by the chi-square distribution with $k - 1$ degrees of freedom. We are now in a position to study the quality of this approximation in the case of relatively small values of $n_1, \dots, n_k$. It is important to remark that this approximation is not very good in the most important region between the 0.95 and the 0.99 quantile (Figure 3).

```
In[84]:= nlist = {2, 3, 4};
k = Length[nlist];
plot1 = PlotCDFG[KruskalWallis[nlist], DisplayFunction -> Identity];
plot2 = Plot[CDF[ChiSquareDistribution[k - 1], x],
  {x, 0, 8}, DisplayFunction -> Identity];
fig3 = Show[{plot1, plot2}, PlotRange -> All,
  DisplayFunction -> $DisplayFunction, Axes -> True]
```

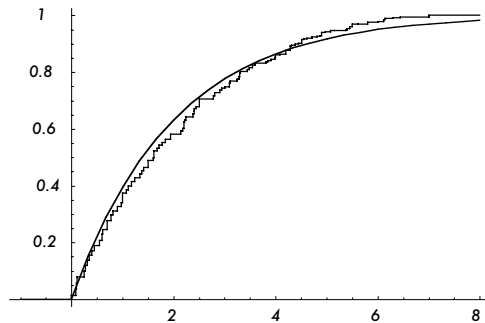


Figure 3.

□ Ties

If ties occur, we assign to tied observations the same midrank and calculate the sum $R_1, \dots, R_k$ of the midranks of the $X_{1\bullet}$'s, $\dots$, $X_{k\bullet}$'s in the combined ordered arrangement of these k samples. We reject the null hypotheses H_0 , if the modified Kruskal–Wallis statistics

$$\text{KruskalWallisTies}[\text{dlist}, \text{nlist}] = \frac{1}{\Delta} \left[\frac{12}{n(n+1)} \sum_{i=1}^k \frac{1}{n_i} R_i^2 - 3(n+1) \right]$$

with

$$\Delta = 1 - \frac{1}{n^3 - n} \sum_{j=1}^l (d_j^3 - d_j)$$

and $n = n_1 + \dots + n_k$ is too large.

Calculating the generating function $G[\text{KruskalWallisTies}[\text{dlist}, \text{nlist}]]$ of the null distribution of this test statistics, it is again essential to mention that

$$\Pr(\{R_1 = r_1\} \cap \dots \cap \{R_k = r_k\}) = \frac{Q[\text{dlist}, m[\text{dlist}], \text{nlist}, \text{rlist}]}{\text{Multinomial}[n_1, \dots, n_k]}.$$

Thus, we get this generating function $G[\text{KruskalWallisTies}[\text{dlist}, \text{nlist}]]$ by applying the substitution

$$s1 = \left\{ y[i]^r \rightarrow t^{\wedge} \left(\frac{r^2}{n_i} \right) \mid i = 1, \dots, k \right\}$$

to

$$\frac{q[\text{dlist}, m[\text{dlist}], \text{nlist}]}{\text{Multinomial}[n_1, \dots, n_k]}$$

and the substitution

$$s2 = t^x \rightarrow t^{\wedge} \left(\frac{1}{\Delta} \left[\frac{12x}{n(n+1)} - 3(n+1) \right] \right)$$

to the result of the first substitution.

```
G[KruskalWallisTies[dlist_, nlist_]] :=
Module[{d, n, k, a, b, Delta, s1, s2},
d = Apply[Plus, dlist];
n = Apply[Plus, nlist];
k = Length[nlist];
a = 12 / (n (n + 1));
b = 3 (n + 1);
Delta = 1 - Apply[Plus, (dlist^3 - dlist)] / (n^3 - n);
s1 = Table[y[i]^r_ -> t^(r^2/nlist[[i]]), {i, 1, k}];
s2 = t^x_ -> t^((a x - b) / Delta);
If[d == n, Apart[
(q[dlist, m[dlist], nlist] /. s1 /. s2) / Apply[Multinomial, nlist]],
Print["d1+...+dl ≠ n1+...+nk"]; Abort[]]
```

```
In[90]:= G[KruskalWallisTies[{2, 2, 3}, {3, 2, 2}]]
```

$$\text{Out[90]} = \frac{2t^{7/50}}{35} + \frac{2t^{33/100}}{35} + \frac{8t^{19/50}}{35} + \frac{2t^{63/50}}{35} + \frac{4t^{153/100}}{35} + \frac{4t^{209/100}}{35} + \frac{t^{69/25}}{35} + \frac{2t^{293/100}}{35} + \frac{4t^{77/25}}{35} + \frac{2t^{17/4}}{35} + \frac{2t^{461/100}}{35} + \frac{t^5}{35} + \frac{2t^{126/25}}{105} + \frac{t^6}{105}$$

Again we have, for example,

```
In[91]:= TableCDFG[KruskalWallisTies[{2, 2, 3}, {3, 2, 2}], 1, 6]
```

```
Out[91]//TableForm=
```

x	CDF[x]
1.26	0.342857
1.53	0.4
2.09	0.514286
2.76	0.628571
2.93	0.657143
3.08	0.714286
4.25	0.828571
4.61	0.885714
5.	0.942857
5.04	0.971429

Lehmann [2] mentions that for large values of $n_1, \dots, n_k$ it is possible to approximate the null distribution of the test statistics `KruskalWallisTies[dlist, nlist]` by the chi-square distribution with $k - 1$ degrees of freedom. By experimenting we can investigate the quality of this approximation graphically in the case of small values of $n_1, \dots, n_k$. It is again important to remark that this approximation is not very good in the most important region between the 0.95 and the 0.99 quantile (Figure 4).

```
In[92]:= dlist = {3, 2, 1, 3, 2};
nlist = {4, 3, 4};
k = Length[nlist];
plot1 = PlotCDFG[
  KruskalWallisTies[dlist, nlist], DisplayFunction -> Identity];
plot2 = Plot[CDF[ChiSquareDistribution[k - 1], x],
  {x, 0, 10}, DisplayFunction -> Identity];
fig4 = Show[{plot1, plot2}, PlotRange -> All,
  DisplayFunction -> $DisplayFunction, Axes -> True]
```

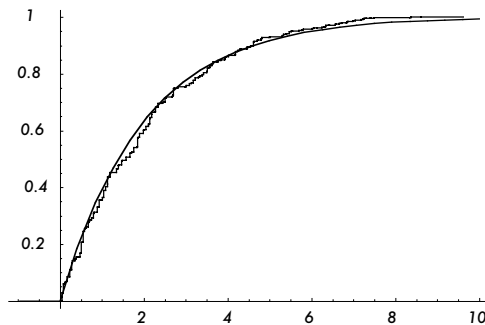
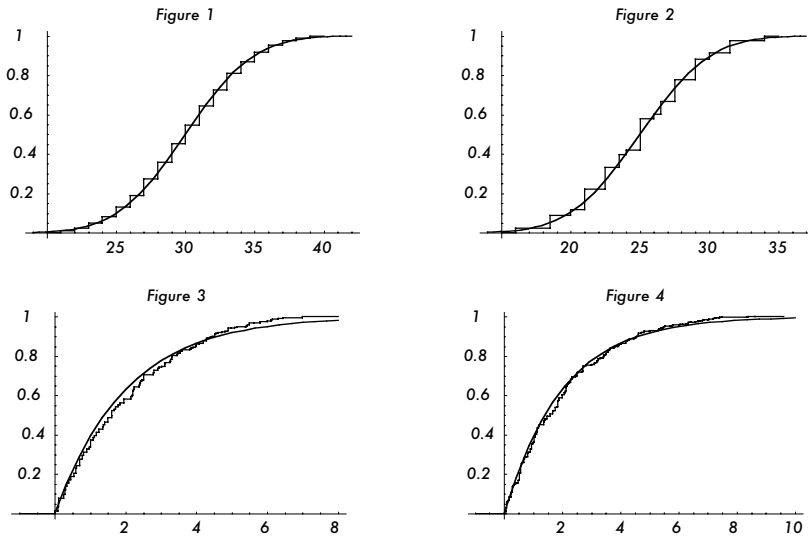


Figure 4.

The following picture shows the approximation of the null distribution of the Wilcoxon rank sum statistics with (Figure 2) and without (Figure 1) ties by an appropriate normal distribution and the approximation of the null distribution of the Kruskal–Wallis statistics with (Figure 4) and without (Figure 3) ties by an appropriate chi-square distribution.



■ Back to our Introductory Example

We are now able to calculate the exact value of the 0.95 quantile $\kappa_{0.95}$ of our test statistics.

```
In[103]:= dlist = {2, 2, 4, 2, 2, 2, 1};
          nlist = {5, 5, 5};
          QTLG[KruskalWallisTies[dlist, nlist], 0.95] // N

Out[105]:= 5.65651
```

Because the value of our test statistics is 5.85062, we will reject the null hypothesis. Approximating the null distribution of our test statistics by the chi-square distribution with 2 degrees of freedom—the 0.95 quantile of this distribution is 5.99145—we would not be able to reject the null hypothesis. This example shows that it is important to work with the exact values of quantiles and not with their approximations.

■ Conclusion

We have shown how the method of generating functions can be used to investigate the null distribution of the well-known Wilcoxon rank sum statistics and the

Kruskal–Wallis statistics (with and without ties). The proposed method is not confined to these test statistics; in contrary, the reader should be able now to apply this method to investigate nearly every test statistic based on ranks.

■ Acknowledgments

I would like to thank the referee for valuable suggestions.

■ References

- [1] *Statistica*, St. Paul, MN: Assessment Systems Corporation, (May 2005) www.assess.com/Software/StatMain.htm.
- [2] E. L. Lehmann, *Nonparametrics: Statistical Methods Based on Ranks*, New York: McGraw-Hill, 1975.
- [3] H. Chernoff and I. R. Savage, "Asymptotic Normality and Efficiency of Certain Nonparametric Test Statistics," *Annals of Mathematical Statistics*, **29**, 1958 pp. 972–994.
- [4] G. E. Noether, *Introduction to Statistics: The Nonparametric Way*, New York: Springer-Verlag, 1991.
- [5] S. Kokoska and C. Nevison, *Statistical Tables and Formulae*, New York: Springer-Verlag, 1989.
- [6] P. Mitic, "Critical Values for the Wilcoxon Signed Rank Statistic," *The Mathematica Journal*, **6**(3), 1996 pp. 73–77.
- [7] W. B. Davenport, *Probability and Random Processes*, New York: McGraw-Hill, 1970.
- [8] V. K. Rohatgi, *An Introduction to Probability Theory and Mathematical Statistics*, New York: John Wiley & Sons, 1976.
- [9] N. L. Johnson and S. Kotz, *Urn Models and Their Application*, New York: John Wiley & Sons, 1977.
- [10] M. Hall Jr., *Combinatorial Theory*, New York: John Wiley & Sons, 1986.

About the Author

Peter Weiß studied mathematics at the University of Innsbruck. Since 1973 he has been a professor for probability and mathematical statistics at the Johannes Kepler University in Linz. His current areas of interest are point-processes, random walk on random structures, and statistical data analysis for industrial applications. In 1990 he founded the new curriculum "Mechatronics," which is a combination of mechanical engineering, electrical engineering, and computer science.

Peter Weiß

*Institute of Mathematics
Johannes Kepler University
Linz, Austria
peter.weiss@jk.uni-linz.ac.at*

On the Numerical Accuracy of Mathematica 5.0 for Doing Linear and Nonlinear Regression

Marc Nerlove

This article presents the results of performing the linear and nonlinear regressions used as benchmarks by the National Institute of Standards and Technology (NIST) with *Mathematica* 5.0. The performance is nearly flawless.

“Here is the guess of their true strength and forces
By diligent discovery;...”
King Lear, V. i. 52

■ 1. Introduction

There is a small but growing literature on the numerical accuracy of various computer programming packages. A major contributor to the assessment of the accuracy of econometric software, as well as statistical add-ons to third-level programming languages widely used by econometricians, such as GAUSS, MATLAB, Maple, and *Mathematica*, has been Bruce D. McCullough ([1] and the references cited therein).

McCullough and Wilson [2, 27] assessed the reliability of the statistical procedures in Excel 97 for estimation of both linear and nonlinear regression, random number generation, and the calculation of cumulative distributions and concluded that “Excel’s performance in all three areas is found to be inadequate. Persons desiring to conduct analyses of statistical data are advised not to use Excel.” McCullough made a complete assessment of all of *Mathematica* 4.0’s then extant statistical functions [3]. In 2000, I made a similar assessment for ordinary least squares (OLS) linear regression analysis using Excel 2000 and the Numeri-

cal Algorithms Group's (NAG's) Statistical Add-Ins for Excel software (1). I found that the "NAG-enhanced" version of Excel did not do significantly better than the non-enhanced version. In the same paper, I also compared the performance of *Mathematica* 4.1.

The basis for McCullough's assessments as well as mine were comparisons with the benchmark results for the Statistical Reference Datasets (StRD) that are available from the NIST (2).

My comparison of Excel 2000 with the benchmark linear regression analyses produced results that did not differ significantly from those McCullough obtained for Excel 97. Notwithstanding the excellent reputation of NAG's software, the performance of the Excel add-on is not superior to Excel. For cases in which Excel performs well, the NAG add-on also does well, but when Excel performs unsatisfactorily, the NAG add-on does poorly as well. However, often the results differ from the certified results in a different direction from those obtained directly in Excel. In the case of linear regression, on the other hand, Excel on its own does not do much worse than many other packages analyzed by McCullough and others. My comparisons at the time suggested that McCullough's judgment, at least with respect to linear regression, may have been unduly critical; however, I do agree that the statistical functions in Excel should not be used with or without NAG enhancement.

Vinod [4, 211] undertook a careful analysis of the statistical functions in GAUSS for Windows Version 3.2.27, also using comparisons with the StRD. He found "... that the algorithms used by GAUSS sometimes fail rather badly to provide accurate results. We blame both the algorithms used and the language itself." I have not yet undertaken a similar reassessment of the more recent and allegedly significantly improved versions of GAUSS.

McCullough [5, 152] applied similar methodology to SAS, SPSS, and S-Plus, which I call fourth-level programming languages or statistical packages. With respect to univariate statistics, analysis of variance, and linear regression, McCullough reported that "All [SAS] results are quite accurate," in terms of six digits of accuracy or better, although SAS failed to deliver any solution in one of the worst cases. Even though SAS failed to achieve convergence in a number of cases, the results for nonlinear regression were similar. The performance of all three packages with respect to random number generation and the calculation of probabilities for statistical distributions was notably poorer than for OLS linear regression.

McCullough [3] analyzed the accuracy of *Mathematica* 4.0's statistics add-on package (3). *Mathematica* is not restricted as to precision; however, precision may be specified. The default is to use machine arithmetic, which is typically 16 digits of precision in internal calculations (a typical value of `$MachinePrecision` is $53 \log_{10} 2$ or approximately 16). Of course this does not guarantee 16 digits of precision in the final result. Note that accuracy and precision are not the same as defined in *Mathematica*. A higher level of precision may be attained by setting `$MinPrecision` above 16 or by setting `WorkingPrecision` $\rightarrow n$, where n is a number greater than `MachinePrecision`. Only a small number of *Mathematica*

functions support the `WorkingPrecision` option. Moreover, when software other than *Mathematica* is used, increasing the level of precision in this way may not be possible. This is because within the code the computation has been carried out in floating-point notation. No matter how many degrees of arbitrary precision are requested, the result is still contaminated with the errors from floating-point arithmetic. Arbitrary precision only works well when the computation has been carried out uncontaminated with floating-point error. In *Mathematica* it is possible to carry out the computation with numbers expressed in rational form, which yields exact results, but this still will not avoid the problem of inverting an ill-conditioned matrix.

At default precision, *Mathematica* is quite accurate although McCullough obtained no solution in one StRD case of OLS linear regression and one in an StRD case of nonlinear regression. When `$MinPrecision` is raised to 20 digits, McCullough reports that astonishing, almost perfect, accuracy, as compared with the NIST benchmark results, is obtained in all cases considered. Actually, *Mathematica*'s default level of precision gives exact matches to the benchmark results in most cases, but the results *Mathematica* displays are rounded to six places and thus may not look quite the same. To display the results to more places, use `NumberForm`. The increase in computational time using a higher level of precision than default in direct, and therefore symbolically pure, computation of the OLS estimates appears to be negligible no matter how high the level of precision is specified. Although it is certainly true that increased accuracy may not matter for the conclusions to be drawn from a statistical analysis, it may also happen that rounding errors in the course of an otherwise straightforward calculation may result in wildly different results or that no solution is found. If the cost of obtaining accurate results even in problematic cases is not great, it is certainly prudent to obtain what increased accuracy may be possible. If a preprogrammed package is used, as it is in this article, it may not be possible to control for rounding error because of the original coding.

In this article, I discuss the details of my own comparisons of *Mathematica* 5.0's standard statistical add-on package. McCullough employs summary measures, which must of necessity leave out a lot of relevant detail in the case of many parameters, but I present a full range of results. A separate section deals with the more computationally difficult benchmark cases for nonlinear regression.

In the next section, I describe the NIST StRD benchmarks, particularly the 11 designed for benchmarking OLS linear regression (i.e., linear in the parameters to be estimated, so polynomial regression is not excluded) and the 27 nonlinear benchmark datasets. I also discuss the way in which the NIST certified results have been obtained for the linear regression StRD set. In Section 3, I discuss the general problems of nonlinear regression analysis and the difficulty obtaining appropriate significance tests short of computing the parameter estimates by maximum likelihood (ML). In Section 4, I summarize the results of using Version 5.0 of *Mathematica*'s `Statistics`LinearRegression`` standard add-on package and compare these with the StRD benchmark results. Essentially the same results may be obtained directly by writing out the standard textbook formulae in *Mathematica*, although these formulae are not the best way of carry-

ing out the requisite numerical calculations. Details of the numerical results are presented in Appendices A and B, in which tabular comparisons are made of the NIST StRD benchmark results with those obtained using *Mathematica*'s `Statistics'LinearRegression` standard add-on package and those using the `NonlinearRegress` option in the `Statistics'NonlinearFit` standard add-on package. The appendices are part of the electronic version of this article, but are not included in the printed version.

■ 2. The NIST StRD Datasets and Certified Results for Linear and Nonlinear Regression

The NIST describes the statistical benchmark program on their website as follows.

“Currently datasets and certified values are provided for assessing the accuracy of software for univariate statistics, linear regression, nonlinear regression, and analysis of variance. The collection includes both generated and ‘real-world’ data of varying levels of difficulty. Generated datasets are designed to challenge specific computations. These include the classic Wampler datasets for testing linear regression algorithms and the Simon & Lesage datasets for testing analysis of variance algorithms. Real-world data include challenging datasets such as the Longley data for linear regression, and more benign datasets such as the Daniel & Wood data for nonlinear regression. Certified values are ‘best-available’ solutions. The certification procedure is described in the web pages for each statistical method.

“Datasets are ordered by level of difficulty (lower, average, and higher). Strictly speaking the level of difficulty of a dataset depends on the algorithm. These levels are merely provided as rough guidance for the user. Producing correct results on all datasets of higher difficulty does not imply that your software will pass all datasets of average or even lower difficulty. Similarly, producing correct results for all datasets in this collection does not imply that your software will do the same for your particular dataset. It will, however, provide some degree of assurance, in the sense that your package provides correct results for datasets known to yield incorrect results for some software” (2).

□ (a) The NIST StRD Datasets for Linear Regression

The particular sets for linear regression analysis are summarized in Table 1. The problem of linear regression and the methods employed by the NIST, as well as *Mathematica*, are discussed in Section 3 (a).

Dataset Name	Level of Difficulty	Model Class	Number of Parameters	Number of Observations	Source
Norris [6]	Lower	Linear	2	36	Observed
Pontius [7]	Lower	Quadratic	3	40	Observed
NoInt1 [8]	Average	Linear	1	11	Generated
NoInt2 [8]	Average	Linear	1	3	Generated
Filip [9]	Higher	Polynomial	11	82	Observed
Longley [10]	Higher	Multilinear	7	16	Observed
Wampler1 [11]	Higher	Polynomial	6	21	Generated
Wampler2 [11]	Higher	Polynomial	6	21	Generated
Wampler3 [11]	Higher	Polynomial	6	21	Generated
Wampler4 [11]	Higher	Polynomial	6	21	Generated
Wampler5 [11]	Higher	Polynomial	6	21	Generated

Source: www.itl.nist.gov/div898/strd/ils/ils.shtml

Table 1. StRD benchmark datasets for linear regression.

The following edited comments from the NIST website describe these datasets in more detail.

“Both generated and ‘real-world’ data are included. Generated datasets challenge specific computations and include the Wampler data developed at NIST (formerly NBS) in the early 1970s. Real-world data include the challenging Longley data, as well as more benign datasets from our statistical consulting work at NIST.

“... Two datasets are included for fitting a line through the origin. We have encountered codes that produce negative R-squared and incorrect F-statistics for these datasets. Therefore, we assign them an ‘average’ level of difficulty. Finally, several datasets of higher level of difficulty are provided. These datasets are multicollinear. They include the Longley data and several NIST datasets developed by Wampler.

“... Certified values are provided for the parameter estimates, their standard deviations, the residual standard deviation, R-squared, and the standard ANOVA table for linear regression. Certified values are quoted to 16 significant digits and are accurate up to the last digit, due to possible truncation errors.

“... If your code fails to produce correct results for a dataset of higher level of difficulty, one possible remedy is to center the data and rerun the code. Centering the data, that is, subtracting the mean for each predictor variable, reduces the degree of multicollinearity. The code may produce correct results for the centered data. You can judge this by comparing predicted values from the fit of centered data with those from the certified fit.”

(www.itl.nist.gov/div898/strd/ils/ils_info.shtml).

□ **(b) The NIST StRD Datasets for Nonlinear Regression**

The particular sets for nonlinear regression analysis are summarized in Table 2. The problem of nonlinear regression and the methods employed by the NIST, as well as *Mathematica*, are discussed in Section 3 (b).

Dataset Name	Level of Difficulty	Model Classification	Number of Parameters	Number of Observations	Source
Misra1a [12]	Lower	Exponential	2	14	Observed
Chwirut2 [13]	Lower	Exponential	3	54	Observed
Chwirut1 [13]	Lower	Exponential	3	214	Observed
Lanczos3 [14]	Lower	Exponential	6	24	Generated
Gauss1 [15]	Lower	Exponential	8	250	Generated
Gauss2 [15]	Lower	Exponential	8	250	Generated
DanWood [16]	Lower	Miscellaneous	2	6	Observed
Misra1b [12]	Lower	Miscellaneous	2	14	Observed
Kirby2 [17]	Average	Rational	5	151	Observed
Hahn1 [18]	Average	Rational	7	236	Observed
Nelson [19]	Average	Exponential	3	128	Observed
MGH17 [20]	Average	Exponential	5	33	Generated
Lanczos1 [14]	Average	Exponential	6	24	Generated
Lanczos2 [14]	Average	Exponential	6	24	Generated
Gauss3 [15]	Average	Exponential	8	250	Generated
Misra1c [12]	Average	Miscellaneous	2	14	Observed
Misra1d [12]	Average	Miscellaneous	2	14	Observed
Roszman1 [21]	Average	Miscellaneous	4	25	Observed
ENSO [22]	Average	Miscellaneous	9	168	Observed
MGH09 [23]	Higher	Rational	4	11	Generated
Thurber [24]	Higher	Rational	7	37	Observed
BoxBOD [25]	Higher	Exponential	2	6	Observed
Rat42 [26]	Higher	Exponential	3	9	Observed
MGH10 [27]	Higher	Exponential	3	16	Generated
Eckerle4 [28]	Higher	Exponential	3	35	Observed
Rat43 [26]	Higher	Exponential	4	15	Observed
Bennett5 [29]	Higher	Miscellaneous	3	154	Observed

Source: www.itl.nist.gov/div898/strd/nls/nls_main.shtml

Table 2. StRD benchmark datasets for nonlinear regression.

The following edited comments from the NIST website describe these datasets in more detail.

“... Hiebert [30] notes that ‘testing to find a “best” code is an all but impossible task and very dependent on the definition of “best.”’ Whatever other criteria are used, the test procedure should certainly attempt to measure the ability of the code to find solutions. But nonlinear least squares regression problems are intrinsically hard, and *it is generally possible to find a dataset that will defeat even the most robust codes. So most evaluations of nonlinear least squares software should also include a measure of the reliability of the code, that is, whether the code correctly recognizes when it has (or has not) found a solution.* The datasets provided here are particularly well suited for such testing of robustness and reliability. [Emphasis added.]

“... both generated and ‘real-world’ nonlinear least squares problems of varying levels of difficulty [are included]. The generated datasets are designed to challenge specific computations. Real-world data include challenging datasets such as the Thurber problem, and more benign datasets such as Misra1a. The certified values are ‘best-available’ solutions, obtained using 128-bit precision and confirmed by at least two different algorithms and software packages using analytic derivatives.

“... For some of these test problems, however, it is unreasonable to expect the correct solution from a nonlinear least squares procedure when finite difference derivatives are used.... These difficult problems are impossible to solve correctly when the matrix of predictor variables is only *approximate* because the user did not supply analytic derivatives.

“... the datasets have been ordered by level of difficulty (lower, average, and higher). This ordering is meant to provide rough guidance for the user. Producing correct results on all datasets of higher difficulty does not imply that your software will correctly solve all datasets of average or even lower difficulty. Similarly, producing correct results for all datasets in this collection does not imply that your software will do the same for your own particular dataset. It will, however, provide some degree of assurance, in the sense that your package provides correct results for datasets known to yield incorrect results for some software.

“The robustness and reliability of nonlinear least squares software depends on the algorithm used and how it is implemented, as well as on the characteristics of the actual problem being solved. Nonlinear least squares solvers are particularly sensitive to the starting values provided for a given problem. For this reason, we provide three sets of starting values for each problem: the first is relatively far from the final solution; the second relatively close; and the third is the actual certified solution.

“... sometimes good starting values are not available. For testing purposes, therefore, it is of interest to see how a code will perform

when the starting values are not close to the solution, even though such starting values might be ridiculously bad from a practical standpoint. In general, it can be expected that a particular code will fail more often from the starting values far from the solution than from the starting values that are relatively close. How serious it is that a code fails when using starting values far from the solution will depend on the types of problems for which the code will be employed.” (www.itl.nist.gov/div898/strd/nls/nls_info.shtml).

In my assessment of *Mathematica* 5.0, I use only the NIST starting values, which are different from the certified results, except in one instance. This is the MGH10 dataset, for which *Mathematica* gives no results for both the first and second starting values. When the certified results are used as the starting value, *Mathematica* crashes. The problem of choosing good starting values is central to nonlinear regression. As discussed later, I believe that such a choice should not be made automatically, but rather involve some prior investigation.

■ 3. Methods for Linear and Nonlinear Regression

□ (a) Linear Regression

The general statistical model assumed for the linear least-squares regression problems is

$$y = X\beta + \epsilon,$$

where y denotes the response (dependent) variable, β denotes the vector of p unknown parameters to be estimated, n is the number of observations, and X denotes the n by p matrix of predictor (independent) variables. The specific functional form for each dataset is different from set to set and is specified in the certified results for that set, summarized in Appendix A. Unless the constant term in the regression is suppressed, X is assumed to contain a column of ones, the coefficient of which is the nonzero intercept. If we want to make inferences about β or how well the relationship fits the data, we have to make some assumptions about the joint distribution of X and ϵ . The usual minimal assumptions are that ϵ has a distribution with variance-covariance matrix $\sigma^2 I$ independently of X . In this case, the least-squares estimates of β ,

$$b = (X'X)^{-1} X'y,$$

have certain optimal properties, and the estimated standard errors of b are

$$se = \sqrt{s^2(X'X)^{-1}},$$

where

$$s^2 = \frac{e'e}{n-k}$$

is an estimate of σ^2 and

$$e = y - Xb$$

are the calculated residuals from the regression. The measure R^2 of goodness-of-fit is defined as

$$R^2 = 1 - \frac{e'e}{y'y - n\bar{y}}$$

when an intercept (frequently represented by including a column of ones in the matrix X) is included and defined as

$$R^2 = 1 - \frac{e'e}{y'y}$$

when no intercept is included (e.g., NoInt1 and NoInt2 [8]). Here is what the NIST says about the method employed for linear regression.

“For all datasets, multiple precision calculations (accurate to 500 digits) were made using the preprocessor and FORTRAN subroutine package of Bailey (1995). Data were read in exactly as multiple precision numbers and all calculations were made with this very high precision. The results were output in multiple precision, and only then rounded to fifteen significant digits. These multiple precision results are an idealization. They represent what would be achieved if calculations were made without roundoff or other errors. Any typical numerical algorithm (i.e., not implemented in multiple precision) will introduce computational inaccuracies, and will produce results which differ slightly from these certified values.” (See the certification methodology description for each dataset at www.itl.nist.gov/div898/strd/lls/lls.shtml, accessed 07/12/03.)

Although not generally advisable, it is perfectly possible to simply write out these equations in *Mathematica* and solve a particular problem, since standard methods for solving for inverses of well-conditioned matrices may not give accurate answers for ill-conditioned matrices. This was what, in fact, went wrong with most of the extant computer packages for doing ordinary least-squares regression when Longley [10] did his famous study. *Mathematica*, however, is very sophisticated about the way it inverts matrices and, indeed, gets this problem right, albeit with a warning message in the program exhibited for the computation of `invX1`.

Daniel Lichtblau comments on the difference between these two methods of doing OLS linear regression as follows: “What we [*Mathematica*] actually work with [in the add-on package, `Statistics'LinearRegression'`] are singular values. In effect, this is a method to compute a pseudo-inverse (which we do not actually do, unless necessitated by the various report option settings). In contrast, matrix inversion will be done using Gaussian elimination, with appropriate warnings if the matrix is found to be ill conditioned.”

Here is my “direct” program.

```
longley = Import["Longley.txt", "Table"];
data = Rest[longley];
{n, k} = Dimensions[data];
y = Map[First, data];
X = Map[Rest, data];
X1 = Map[Prepend[#, 1] &, X];
invX1 = Inverse[Transpose[X1].X1];
b = invX1.(y.X1);
res = y - b.Transpose[X1];
s2 = res.res / (n - k);
se = Sqrt[s2*Tr[invX1, List]];
ȳ = Apply[Plus, y] / n;
R2 = 1 - (res.res) / ((y - ȳ).(y - ȳ));
```

The reader, of course, will have to fill in the exact location of the Longley.txt file, which is copied as a .txt file in Appendix C.

The *Mathematica* parameter estimates and standard errors compared with the benchmark results are:

	<i>Mathematica</i>	Benchmark	<i>Mathematica</i>	Benchmark
Parameters	Estimate	Estimate	SE	SE
β_0	-3.48226×10^6	-3.48226×10^6	890420.	890420.
β_1	15.0619	15.0619	84.9149	84.9149
β_2	-0.0358192	-0.0358192	0.033491	0.033491
β_3	-2.02023	-2.02023	0.4884	0.4884
β_4	-1.03323	-1.03323	0.214274	0.214274
β_5	-0.0511041	-0.0511041	0.226073	0.226073
β_6	1829.15	1829.15	455.478	455.478

And of R^2 and s^2 :

Parameters	<i>Mathematica</i> Values	Benchmark Values
s^2	92936.	92936.
R^2	0.995479	0.995479

If I were to exhibit the results to 15 digits, the *Mathematica* and benchmark results would also coincide exactly. Section 3.1.4 of *The Mathematica Book* [31] gives a good discussion of the way in which *Mathematica* handles the problem of numerical precision in general and floating-point arithmetic in particular. So when doing linear least-squares regression it does not appear to matter whether I use my own code or *Mathematica*'s `Regress` function.

I have not been able to find documentation as to exactly how *Mathematica* performs a linear least-squares regression with its `Regress` function, other than the full code for `LinearRegression` in `AddOns/StandardPackages/Statistics'LinearRegression'`. (See Daniel Lichtblau's earlier comments.) Needless to say, it is not so easy for a nonprogrammer to read this code. It

appears that *Mathematica* uses the same computationally efficient and accurate methods to solve the usual least-squares normal equations as it does to solve any system of linear equations or to invert a numerically specified square matrix. See Section A.9.4 of *The Mathematica Book* [31].

Comparisons of the results from `Regress` with the NIST benchmarks for each of the 11 NIST SdRD linear regression examples are presented in summary form in Appendix A. The results presented for *Mathematica* are simply those *Mathematica* prints out. The reader, however, should be aware that these are not the actual results of *Mathematica*'s computation, which are all carried out to 16 digits of precision, but rather a rounded form for ease of visual presentation. The full 16 digits can easily be obtained by using `NumberForm[expr, {n, f}]`, which prints approximate real numbers having n digits, with f digits to the right of the decimal point.

□ (b) Nonlinear Regression

Following the NIST description, the statistical model assumed for nonlinear regression in the benchmark calculations is the so-called generic univariate case:

$$y = f(x; \beta) + \epsilon,$$

where y denotes the response (dependent) variable, x denotes the predictor (independent) variables, and β (unsubscripted) the p unknown parameters to be estimated. The specific functional forms assumed differ from one benchmark case to the next. They are reported in Appendix B, in which comparisons between the results obtained by *Mathematica* and those obtained by the NIST are presented. The NIST methodology is described as follows.

“The certified values for the nonlinear least-squares regression problems were obtained using 128-bit precision with the reported results confirmed by at least two different algorithms and software packages using analytic derivatives.

“The certified values for the estimates, $b = (b_1, b_2, \dots, b_p)$ of the true model parameters β are those produced by the smallest sum of squares, i.e.,

$$b = \operatorname{argmin}_{\beta} \left\{ \sum_{i=1}^n [y_i - f(x_i; \beta)]^2 \right\},$$

where n denotes the number of observations. Under the assumption that

$$\epsilon_i \equiv [y_i - f(x_i; \beta)] \sim N(0, \sigma^2),$$

it follows that these are the maximum-likelihood estimators.

“The certified values for the standard deviations of the estimates of the model parameters are the square roots of the diagonal elements of the asymptotic covariance matrix,

$$V = s^2 [\mathcal{J}' \mathcal{J}]^{-1},$$

where

$$s = \sqrt{\frac{\min_{\beta} \left\{ \sum_{i=1}^n [y_i - f(x_i; \beta)]^2 \right\}}{n - p}} = \sqrt{\frac{\min_{\beta} \left\{ \sum_{i=1}^n [y_i - f(x_i; b)]^2 \right\}}{n - p}},$$

$\mathcal{J}$ denotes the Jacobian matrix with ij th element

$$\mathcal{J}_{ij} = \frac{\partial f(x_i; \beta)}{\partial \beta_j}, \quad i = 1, 2, \dots, n; \quad j = 1, 2, \dots, p,$$

evaluated at the current values of the parameters $b_1, b_2, \dots, b_p$, and n and p denote the number of observations and the number of parameters, respectively.

“The certified value of the residual sum of squares is defined by

$$SS = \min_{\beta} \left\{ \sum_{i=1}^n [y_i - f(x_i; \beta)]^2 \right\} = \min_{\beta} \left\{ \sum_{i=1}^n [y_i - f(x_i; b)]^2 \right\}$$

where n denotes the number of observations.”

The benchmark estimate of σ^2 is thus $SS / (n - p)$ (www.itl.nist.gov/div898/strd/nls/nls_info.shtml).

Under the assumption $\epsilon_i \equiv [y_i - f(x_i; \beta)] \sim N(0, \sigma^2)$, it is thus possible to derive all of these estimates by maximum likelihood. In this case, the standard errors of the estimates are obtained asymptotically as the square roots of the diagonal elements of the inverse of the information matrix (the Hessian of the likelihood function evaluated at the maximizing point). (See the standard econometric presentations of nonlinear regression methods in Davidson and MacKinnon [32] and Greene [33].) Indeed, it is in some ways a more general method of analysis, since ML may be applied in the case of more general models.

Before proceeding, it is worth noting a few limitations of the standard univariate nonlinear regression model, $y = f(x; \beta) + \epsilon$. According to Davidson and MacKinnon [32, 42], “The feature that distinguishes regression models from all other statistical models is that the only way in which randomness affects the dependent variable is through an *additive* error term or disturbance.” However, this may be less of a restriction than you might at first think. For example, suppose a multiplicative disturbance $y = f(x; \beta) * \epsilon$; then a simple logarithmic transformation suffices to return things to additivity: $\text{Log}[y] = \text{Log}[f(x; \beta)] + \text{Log}[\epsilon]$. Of course any distributional assumptions we may have made concerning ϵ may have to be modified. Independence of the regressors, x , and the disturbance, or at least lack of correlation, is an assumption common to both linear and nonlinear regression. A problem arises, however, if the appropriate transformation depends on parame-

ters to be estimated. For example, a common generalization is a transformation of the dependent variable involving another parameter to be estimated:

$$g(y; \theta) = f(x; \beta) + \epsilon.$$

In this case, the distribution of the y s, given the x s, will no longer be the same as the distribution of the ϵ s, and least squares may be problematic and generally will no longer yield the same estimates as maximum likelihood. Maximum likelihood is still an effective method, however, since the distribution of the y s, given the x s, may still be found from the distribution of the ϵ s by multiplying by the Jacobian of the transformation, $g(y; \theta)$. A more serious limitation is the necessity of specifying the functional form of f . Whereas, in the case of linear regression we hardly ever think about alternative functional forms, except perhaps to worry about interactions, the very fact that we are considering nonlinear regression underscores a concern about the appropriate functional form. In recent years, there has been an efflorescence of work on nonparametric methods explicitly designed to cope with this problem. (See, especially Haerdle [34], and Pagan and Ullah [35]).

There is a extensive literature on nonlinear regression, including standard texts, such as Gallant [36], Bates and Watts [37], and Seber and Wild [38], each with somewhat different emphasis. Most emphasis falls on computational issues and problems of parametric inference. A very good brief expository introduction is Gallant [39].

Most algorithms for computing nonlinear regression parameters are based on Hartley's [40] modified Gauss–Newton method (Hartley and Booker, [41]) or as extended and modified by Marquardt [42] following a suggestion by Levenberg [43]. There is an important difference between maximizing/minimizing functions in general and the problem of finding the minimum of an analytically specified sum of squares (see the previous SS). In general, we can evaluate only the function itself analytically and perhaps its gradient. Using only these two pieces is the basis for the method of steepest descent, sometimes also known as the Gauss–Newton method. But, in the case of nonlinear least squares, we know a great deal more: we know the Hessian matrix. Hartley's insight was to exploit this information explicitly, rather than build it up piecemeal during the iterative process of descent. His modified Gauss–Newton method is called the inverse Hessian method. Levenberg's insight was to recognize that simple steepest descent is much faster and more accurate when you are close to the minimum and to suggest separating the problem of getting into the neighborhood of the minimum from “zipping in.” Marquardt's contribution was an elegant algorithm designed to vary smoothly between the two methods as the minimum point was approached. *Mathematica's* `NonlinearRegress` uses the Levenberg–Marquardt algorithm by default, but also allows you to choose among several other algorithms for function minimization.

In [30, 44], Hiebert did a careful evaluation of existing programs for doing nonlinear least squares and for solving systems of nonlinear equations (a harder problem). One surprising conclusion he drew [30, 15] was that, “On the basis of this testing and these specific implementations of the Levenberg–Marquardt and

the augmented Gauss–Newton methods, one method does not appear to be superior to the other.” Perhaps, for this reason, the NIST uses a variety of methods, each considered appropriate for the problem at hand. However, the NIST is not more explicit about the methods used in specific cases than the statements quoted earlier.

Standard-error calculations and inference for nonlinear least-squares estimates are all based on asymptotic results (Hartley and Booker, [41]; Jennrich, [45]) and deftly summarized by Gallant [39]. These are the basis for the NIST calculations detailed earlier. *Mathematica* does not say how its calculations are carried out in the description of the `NonlinearRegress` function; however, the code is available in the `Statistics`NonlinearRegress`` package, and it appears that the same formulae are used.

Comparisons of the results from `NonlinearRegress` with the NIST benchmarks for each of the 27 NIST StRD nonlinear regression examples with their published starting values are presented in summary form in Appendix B. The NIST actually ran all computations using not only these two starting values but also the final benchmark answers as starting values because some nonlinear regression algorithms have a problem if you start from the right answer. However, in a few cases, I also tried starting at the benchmark answers: in the case of MGH10, as noted earlier, and for those few starting values for which *Mathematica* did not converge: Lanczos2; MGH17; MGH09; as well as MGH10. However, as noted by the NIST, MGH10 is a particularly difficult problem: “This problem was found to be difficult for some very good algorithms. There is a local minimum at (+inf, -14.07 ... , -inf, -inf) with final sum of squares 0.00102734.” [46, 17] Lanczos3 is also a rather peculiar function, but *Mathematica* generally performs quite well for chosen starting values.

The crucial problem of choosing good starting values is discussed extensively by Bates and Watts [37, 72–76], and, more briefly, by Seber and Wild [38, 665–666]. Bates and Watts suggest using a linearized form of the expectation function, while Seber and Wild suggest grid search or randomized starting values. Ratkowsky [26] gives a much more extensive discussion. When only one or two parameters are involved, a graphical approach works wonders. However, when three or more parameters are involved, special techniques are required and the difficulties are formidable. I have been unable to ascertain how the NIST chose their starting values, which are always one quite close to and another far away from the final answer, but *Mathematica* does offer an alternative. Before invoking `NonlinearRegress`, you could use `NMinimize` with no starting values (or some known to cause problems) assumed to find the minimum of the appropriate sum of squares. This is a very elaborate and well-documented method in *Mathematica* for finding global minima. Such more sophisticated methods may not be very efficient compared to the algorithm *Mathematica* employs in `NonlinearRegress`, but may work in such difficult cases. In computing the results reported here, I avoid the problem of choosing starting values by simply using those reported by the NIST.

■ 4. Discussion of the Results Using *Mathematica* 5.0

□ (a) Linear Regression

There is really not much to say about the performance of *Mathematica* in doing linear regression. *Mathematica* is remarkably accurate with only one negligible but puzzling anomaly: In the cases of Wampler1 and Wampler2, both fifth degree polynomials with artificially constructed data, the true SEs are 0, as is the correct estimate of s^2 . However, *Mathematica* delivers very small numbers of the order of 10^{-10} to 10^{-28} . While this is not a practical problem of any significance, some explanation is needed in view of *Mathematica*'s otherwise stellar performance. I would attribute it to *Mathematica*'s use of variable-precision floating-point arithmetic [47], but the matter clearly requires further looking into.

□ (b) Nonlinear Regression

Nonlinear regressions present a greater challenge. In all but five out of 27 difficult nonlinear regression problems, *Mathematica* converged to the benchmark values within the default number of iterations, `MaxIter=100` for both of the two start values given by the NIST. But there were serious problems with five of the problems, including one case in which no convergence was obtained even starting from the final benchmark result. However, in all five cases, informative error messages were given by *Mathematica* and no spurious results were presented. Here is a summary.

MGH17

$$y = \beta_1 + e^{-x\beta_4} \beta_2 + e^{-x\beta_5} \beta_3 + \epsilon$$

As noted earlier, this is an exceptionally difficult problem given the observations on y and x . Start 1 produced an overflow in the computation. There were no problems, however, encountered in obtaining the benchmark results from start 2.

Lanczos2

$$y = e^{-x\beta_2} \beta_1 + e^{-x\beta_4} \beta_3 + e^{-x\beta_6} \beta_5 + \epsilon$$

Note that this is the same functional form as MGH17 without a constant term. In both examples the data has been generated by the NIST. No convergence was obtained for the default number of iterations from start 1 with an error message to that effect. Convergence was, however, obtained for `MaxIter=200` from start 1 and from start 2 for `MaxIter=100` to the correct benchmark results.

MGH09

$$y = \frac{\beta_1 (x^2 + x\beta_2)}{x^2 + x\beta_3 + \beta_4} + \epsilon$$

No convergence was obtained from start 1 even by increasing the number of iterations to 500. The following error message suggests that the function is extremely flat, at least in one direction.

FindFit::"lstol": "The line search decreased the step size to within tolerance specified by AccuracyGoal and PrecisionGoal but was unable to find a sufficient decrease in the norm of the residual. You may need more than MachinePrecision digits of working precision to meet these tolerances."

Convergence to the benchmark results for MaxIter=100 from start 2 was obtained without any problems.

MGH10

$$y = \frac{\beta_1}{1 + e^{\frac{\beta_2}{x + \beta_3}}} + \epsilon$$

No convergence was obtained from start 1 or from start 2 for 10,000 iterations. Starting from the benchmark results produced an abnormal program termination with a message that the fitting algorithm failed.

Bennett5

$$y = \beta_1 (x + \beta_2)^{-\frac{1}{\beta_3}} + \epsilon$$

For start 1, convergence was obtained only for 10,000 iterations. Although the benchmark coefficients and standard errors were obtained, the benchmark estimate of residual variance was not. For start 2, convergence to the benchmark results within 100 iterations was obtained.

Note that *Mathematica*'s NonlinearRegress is extremely fast. The time to run a regression once the data is set up and the command given is imperceptible for the default number of 100 iterations. Even for 10,000 iterations, the running time is barely perceptible on a Dell Workstation, PWS 340, with an Intel® Pentium chip.

Mathematica's NonlinearRegress is an extremely fast and accurate algorithm when it works. When it does not, it delivers error messages rather than spurious results. Presently, there is no provision in the NonlinearFit package for obtaining reasonably good start values.

■ 5. Discussion of the Results Using *Mathematica* 5.1

Darren Glosemeyer

Kernel Developer
Kernel Technology Group
Wolfram Research, Inc.
darreng@wolfram.com

The following results were obtained in *Mathematica* 5.1 on an Intel® Pentium 4 Windows XP computer. Version 5.1 does a little better on some examples than 5.0.

```
In[1]:= $Version
```

```
Out[1]= 5.1 for Microsoft Windows (October 25, 2004)
```

The following comments are for the examples for which problems were noted in 5.0. Results for the other examples in the NIST benchmarks still give correct results in 5.1. For most examples, just the parameter tables are shown; however, the s^2 values were also checked against the benchmark.

For three examples, differences between *Mathematica*'s results and the benchmark result are attributable to numerical error in machine-precision arithmetic, numerical error in the NIST data, or a combination of those two sources. In the other examples, `NonlinearRegress` either gets a correct result with default settings or obtains a correct result with an additional option in 5.1.

```
In[2]:= << Statistics`
```

□ Wampler1

The numerical error in this example is a result of numerical error in machine-precision computation and can be expected.

```
In[3]:= w1 = {{0, 1}, {1, 6}, {2, 63}, {3, 364},
             {4, 1365}, {5, 3906}, {6, 9331}, {7, 19608}, {8, 37449},
             {9, 66430}, {10, 111111}, {11, 177156}, {12, 271453},
             {13, 402234}, {14, 579195}, {15, 813616}, {16, 1118481},
             {17, 1508598}, {18, 2000719}, {19, 2613660}, {20, 3368421}};
```

```
In[4]:= ParameterTable /. Regress[w1, {1, x, x^2, x^3, x^4, x^5},
                                x, RegressionReport -> {ParameterTable}]
```

```
Out[4]//TableForm=
```

	Estimate	SE	TStat	PValue
1	1.	4.98764×10^{-10}	2.00496×10^9	0.
x	1.	5.47711×10^{-10}	1.82578×10^9	0.
x^2	1.	1.80599×10^{-10}	5.53713×10^9	0.
x^3	1.	2.35152×10^{-11}	4.25258×10^{10}	0.
x^4	1.	1.30828×10^{-12}	7.64361×10^{11}	0.
x^5	1.	2.60293×10^{-14}	3.84182×10^{13}	0.

Significance arithmetic (variable-precision arithmetic) can be used to get results of a desired precision or accuracy. `Regress` can be forced to use significance arithmetic by applying `SetPrecision` to the data. Here is an example with precision 20 data with the result displayed at machine-precision. Because there is no random error in the original data, division by 0 will result in error messages when attempting to compute the t statistics and p-values.

```
In[5]:= Off[Power::"infy", Less::"nord"];
N[ParameterTable /. Regress[SetPrecision[w1, 20],
{1, x, x^2, x^3, x^4, x^5}, x, RegressionReport -> {ParameterTable}]]
```

Out[6]//TableForm=

	Estimate	SE	TStat	PValue
1.	1.	0.	ComplexInfinity	Indeterminate
x	1.	0.	ComplexInfinity	Indeterminate
x ²	1.	0.	ComplexInfinity	Indeterminate
x ³	1.	0.	ComplexInfinity	Indeterminate
x ⁴	1.	0.	ComplexInfinity	Indeterminate
x ⁵	1.	0.	ComplexInfinity	Indeterminate

□ Wampler2

In this example, there are two sources of numerical error: rounding in the original data and machine-precision arithmetic. While the NIST data was generated from a polynomial expression, the data is only given to five digits of accuracy, resulting in some roundoff or truncation error in the data.

```
In[7]:= w2 = {{0, 1.}, {1, 1.11111}, {2, 1.24992}, {3, 1.42753}, {4, 1.65984},
{5, 1.96875}, {6, 2.38336}, {7, 2.94117}, {8, 3.68928},
{9, 4.68559}, {10, 6.}, {11, 7.71561}, {12, 9.92992},
{13, 12.75603}, {14, 16.32384}, {15, 20.78125}, {16, 26.29536},
{17, 33.05367}, {18, 41.26528}, {19, 51.16209}, {20, 63.}};
```

```
In[8]:= ParameterTable /. Regress[w2, {1, x, x^2, x^3, x^4, x^5},
x, RegressionReport -> {ParameterTable}]
```

Out[8]//TableForm=

	Estimate	SE	TStat	PValue
1	1.	1.52366×10^{-14}	6.56316×10^{13}	0.
x	0.1	1.67318×10^{-14}	5.97663×10^{12}	0.
x ²	0.01	5.51706×10^{-15}	1.81256×10^{12}	0.
x ³	0.001	7.18357×10^{-16}	1.39207×10^{12}	0.
x ⁴	0.0001	3.99663×10^{-17}	2.50211×10^{12}	0.
x ⁵	0.00001	7.9516×10^{-19}	1.25761×10^{13}	0.

As in the case of Wampler1, more precise results can be obtained by using higher precision input for the regression.

```
In[9]:= Off[Power::"infy", Less::"nord"];
N[ParameterTable /. Regress[SetPrecision[w2, 17],
  {1, x, x^2, x^3, x^4, x^5}, x, RegressionReport -> {ParameterTable}]]
```

Out[10]//TableForm=

	Estimate	SE	TStat	PValue
1.	1.	0.	ComplexInfinity	Indeterminate
x	0.1	0.	ComplexInfinity	Indeterminate
x ²	0.01	0.	ComplexInfinity	Indeterminate
x ³	0.001	0.	ComplexInfinity	Indeterminate
x ⁴	0.0001	0.	ComplexInfinity	Indeterminate
x ⁵	0.00001	0.	ComplexInfinity	Indeterminate

However, increasing the precision of the input cannot make up for the fact that there is some numerical error in the original data. As the precision is increased, the significance of the roundoff or truncation error in the data from the NIST will show up as small deviations from the benchmark results.

```
In[11]:= Off[Power::"infy", Less::"nord"];
N[ParameterTable /. Regress[SetPrecision[w2, 30],
  {1, x, x^2, x^3, x^4, x^5}, x, RegressionReport -> {ParameterTable}]]
```

Out[12]//TableForm=

	Estimate	SE	TStat	PValue
1.	1.	6.38509×10^{-16}	1.56615×10^{15}	1.60271×10^{-220}
x	0.1	7.01171×10^{-16}	1.42618×10^{14}	6.52694×10^{-205}
x ²	0.01	2.312×10^{-16}	4.32526×10^{13}	3.86516×10^{-197}
x ³	0.001	3.01037×10^{-17}	3.32185×10^{13}	2.02606×10^{-195}
x ⁴	0.0001	1.67484×10^{-18}	5.97071×10^{13}	3.06874×10^{-199}
x ⁵	0.00001	3.33223×10^{-20}	3.00099×10^{14}	9.2976×10^{-210}

□ MGH17

Each value in the first set of starting points is about two orders of magnitude from the optimal value. Overflows are quickly encountered using the default Levenberg–Marquardt method. Despite the tough starting values, good results can be obtained using Newton's method instead of the default Levenberg–Marquardt method.

```
In[13]:= mgh17 = {{0., 0.844}, {10., 0.908}, {20., 0.932}, {30., 0.936},
  {40., 0.925}, {50., 0.908}, {60., 0.881}, {70., 0.85}, {80., 0.818},
  {90., 0.784}, {100., 0.751}, {110., 0.718}, {120., 0.685},
  {130., 0.658}, {140., 0.628}, {150., 0.603}, {160., 0.58},
  {170., 0.558}, {180., 0.538}, {190., 0.522}, {200., 0.506},
  {210., 0.49}, {220., 0.478}, {230., 0.467}, {240., 0.457},
  {250., 0.448}, {260., 0.438}, {270., 0.431}, {280., 0.424},
  {290., 0.42}, {300., 0.414}, {310., 0.411}, {320., 0.406}};
```

```
In[14]:= ParameterCITable /. NonlinearRegress[mgh17,  $\beta_1 + \beta_2 e^{-\beta_4 x} + \beta_3 e^{-\beta_5 x}$ ,
      x, {{ $\beta_1$ , 50}, { $\beta_2$ , 150}, { $\beta_3$ , -100}, { $\beta_4$ , 1}, { $\beta_5$ , 2}},
      RegressionReport -> {ParameterCITable}, Method -> Newton]
```

```
Out[14]//TableForm=
```

	Estimate	Asymptotic SE	CI
β_1	0.37541	0.00207232	{0.371165, 0.379655}
β_2	1.93585	0.220317	{1.48455, 2.38715}
β_3	-1.46469	0.221757	{-1.91894, -1.01044}
β_4	0.0128675	0.000448614	{0.0119486, 0.0137865}
β_5	0.0221227	0.00089472	{0.0202899, 0.0239555}

As in 5.0, this example converges to the correct result from the second set of starting values with default options.

□ Lanczos I

With both sets of starting points, this example converges to the expected parameter estimates with default options in 5.1. The SEs and s^2 values are on the same order as the benchmark values, which is reasonable given the magnitudes of the SEs and s^2 and the fact that the benchmark data is given to 13 digits. For the first starting points, the SEs agree with the benchmark in at least the first digit. For the second starting points, the SEs agree with the benchmark in at least the first two digits.

```
In[15]:= l1 = {{0., 2.5134}, {0.05, 2.044333373291},
      {0.1, 1.668404436564}, {0.15, 1.366418021208},
      {0.2, 1.123232487372}, {0.25, 0.9268897180037},
      {0.3, 0.7679338563728}, {0.35, 0.6388775523106},
      {0.4, 0.5337835317402}, {0.45, 0.4479363617347},
      {0.5, 0.377584788435}, {0.55, 0.3197393199326},
      {0.6, 0.2720130773746}, {0.65, 0.2324965529032},
      {0.7, 0.1996589546065}, {0.75, 0.1722704126914},
      {0.8, 0.1493405660168}, {0.85, 0.1300700206922},
      {0.9, 0.1138119324644}, {0.95, 0.1000415587559},
      {1., 0.0883320908454}, {1.05, 0.0783354401935},
      {1.1, 0.06976693743449}, {1.15, 0.06239312536719}};
```

```
In[16]:= res = NonlinearRegress[l1,  $\beta_1 e^{-\beta_2 x} + \beta_3 e^{-\beta_4 x} + \beta_5 e^{-\beta_6 x}$ , x,
      {{ $\beta_1$ , 1.2}, { $\beta_2$ , .3}, { $\beta_3$ , 5.6}, { $\beta_4$ , 5.5}, { $\beta_5$ , 6.5}, { $\beta_6$ , 7.6}},
      RegressionReport → {ParameterCITable, EstimatedVariance}];
ParameterCITable /. res
EstimatedVariance /. res
```

```
Out[17]//TableForm=
```

	Estimate	Asymptotic SE	CI
β_1	0.0951	5.91894×10^{-11}	{0.0951, 0.0951}
β_2	1.	3.04816×10^{-10}	{1., 1.}
β_3	0.8607	1.50628×10^{-10}	{0.8607, 0.8607}
β_4	3.	3.69559×10^{-10}	{3., 3.}
β_5	1.5576	2.08763×10^{-10}	{1.5576, 1.5576}
β_6	5.	1.22684×10^{-10}	{5., 5.}

```
Out[18]=  $9.78511 \times 10^{-27}$ 
```

```
In[19]:= res = NonlinearRegress[l1,  $\beta_1 e^{-\beta_2 x} + \beta_3 e^{-\beta_4 x} + \beta_5 e^{-\beta_6 x}$ , x,
      {{ $\beta_1$ , .5}, { $\beta_2$ , .7}, { $\beta_3$ , 3.6}, { $\beta_4$ , 4.2}, { $\beta_5$ , 4}, { $\beta_6$ , 6.3}},
      RegressionReport → {ParameterCITable, EstimatedVariance}];
ParameterCITable /. res
EstimatedVariance /. res
```

```
Out[20]//TableForm=
```

	Estimate	Asymptotic SE	CI
β_1	0.0951	5.35261×10^{-11}	{0.0951, 0.0951}
β_2	1.	2.75651×10^{-10}	{1., 1.}
β_3	0.8607	1.36216×10^{-10}	{0.8607, 0.8607}
β_4	3.	3.34199×10^{-10}	{3., 3.}
β_5	1.5576	1.88788×10^{-10}	{1.5576, 1.5576}
β_6	5.	1.10946×10^{-10}	{5., 5.}

```
Out[21]=  $8.00218 \times 10^{-27}$ 
```

With an increased AccuracyGoal, s^2 values matching the benchmark to three digits can be obtained.

```
In[22]:= EstimatedVariance /. NonlinearRegress[l1,  $\beta_1 e^{-\beta_2 x} + \beta_3 e^{-\beta_4 x} + \beta_5 e^{-\beta_6 x}$ , x,
      {{ $\beta_1$ , 1.2}, { $\beta_2$ , .3}, { $\beta_3$ , 5.6}, { $\beta_4$ , 5.5}, { $\beta_5$ , 6.5}, { $\beta_6$ , 7.6}},
      RegressionReport → {EstimatedVariance}, AccuracyGoal → 15]
```

```
Out[22]=  $7.94334 \times 10^{-27}$ 
```

```
In[23]:= EstimatedVariance /. NonlinearRegress[l1,  $\beta_1 e^{-\beta_2 x} + \beta_3 e^{-\beta_4 x} + \beta_5 e^{-\beta_6 x}$ , x,
      {{ $\beta_1$ , .5}, { $\beta_2$ , .7}, { $\beta_3$ , 3.6}, { $\beta_4$ , 4.2}, { $\beta_5$ , 4}, { $\beta_6$ , 6.3}},
      RegressionReport → {EstimatedVariance}, AccuracyGoal → 15]
```

```
Out[23]=  $7.94795 \times 10^{-27}$ 
```

□ Lanczos2

With both sets of starting points, this example converges to the expected result with default options in 5.1.

```
In[24]:= l2 = {{0., 2.5134}, {0.05, 2.04433}, {0.1, 1.6684}, {0.15, 1.36642},
             {0.2, 1.12323}, {0.25, 0.92689}, {0.3, 0.767934}, {0.35, 0.638878},
             {0.4, 0.533784}, {0.45, 0.447936}, {0.5, 0.377585},
             {0.55, 0.319739}, {0.6, 0.272013}, {0.65, 0.232497},
             {0.7, 0.199659}, {0.75, 0.17227}, {0.8, 0.149341}, {0.85, 0.13007},
             {0.9, 0.113812}, {0.95, 0.100042}, {1., 0.0883321},
             {1.05, 0.0783354}, {1.1, 0.0697669}, {1.15, 0.0623931}};
```

```
In[25]:= ParameterCITable /. NonlinearRegress[l2,  $\beta_1 e^{-\beta_2 x} + \beta_3 e^{-\beta_4 x} + \beta_5 e^{-\beta_6 x}$ , x,
             {{ $\beta_1$ , 1.2}, { $\beta_2$ , .3}, { $\beta_3$ , 5.6}, { $\beta_4$ , 5.5}, { $\beta_5$ , 6.5}, { $\beta_6$ , 7.6}},
             RegressionReport -> {ParameterCITable}]
```

Out[25]//TableForm=

	Estimate	Asymptotic SE	CI
β_1	0.096251	0.000667706	{0.0948482, 0.0976538}
β_2	1.00573	0.00339896	{0.998592, 1.01287}
β_3	0.864247	0.00171858	{0.860636, 0.867858}
β_4	3.00783	0.0041707	{2.99907, 3.01659}
β_5	1.5529	0.00237444	{1.54791, 1.55789}
β_6	5.00288	0.00139588	{4.99995, 5.00581}

```
In[26]:= ParameterCITable /. NonlinearRegress[l2,  $\beta_1 e^{-\beta_2 x} + \beta_3 e^{-\beta_4 x} + \beta_5 e^{-\beta_6 x}$ , x,
             {{ $\beta_1$ , .5}, { $\beta_2$ , .7}, { $\beta_3$ , 3.6}, { $\beta_4$ , 4.2}, { $\beta_5$ , 4}, { $\beta_6$ , 6.3}},
             RegressionReport -> {ParameterCITable}]
```

Out[26]//TableForm=

	Estimate	Asymptotic SE	CI
β_1	0.096251	0.000667706	{0.0948482, 0.0976538}
β_2	1.00573	0.00339896	{0.998592, 1.01287}
β_3	0.864247	0.00171858	{0.860636, 0.867858}
β_4	3.00783	0.0041707	{2.99907, 3.01659}
β_5	1.5529	0.00237444	{1.54791, 1.55789}
β_6	5.00288	0.00139588	{4.99995, 5.00581}

□ MGH09

This example converges a little faster in 5.1 than in 5.0. Convergence is obtained for the first starting points with MaxIterations→400, as opposed to MaxIterations→600 in 5.0.

```
In[27]:= mgh09 = {{4., 0.1957}, {2., 0.1947}, {1., 0.1735}, {0.5, 0.16},
                  {0.25, 0.0844}, {0.167, 0.0627}, {0.125, 0.0456}, {0.1, 0.0342},
                  {0.0833, 0.0323}, {0.0714, 0.0235}, {0.0625, 0.0246}};
```

```
In[28]:= ParameterCITable /. NonlinearRegress[mgh09,

$$\frac{\beta_1 (x^2 + x \beta_2)}{x^2 + x \beta_3 + \beta_4}, x, \{\{\beta_1, 25\}, \{\beta_2, 39\}, \{\beta_3, 41.5\}, \{\beta_4, 39\}\},$$

RegressionReport → {ParameterCITable}, MaxIterations → 400]
```

Out[28]//TableForm=

	Estimate	Asymptotic SE	CI
β_1	0.192807	0.0114353	{0.165767, 0.219847}
β_2	0.191282	0.196332	{-0.27297, 0.655534}
β_3	0.123057	0.080842	{-0.0681045, 0.314218}
β_4	0.136062	0.0900255	{-0.0768142, 0.348939}

Correct results are obtained from the second set of starting points with the default option settings, as is the case in 5.0.

□ MGH10

Convergence from each set of starting values was obtained with `MaxIterations`→300 in 5.1. This is an improvement for the first set of points, for which `MaxIterations`→500 was used to obtain the benchmark result in 5.0.

```
In[29]:= mgh10 = {{50., 34780.}, {55., 28610.}, {60., 23650.}, {65., 19630.},
{70., 16370.}, {75., 13720.}, {80., 11540.}, {85., 9744.},
{90., 8261.}, {95., 7030.}, {100., 6005.}, {105., 5147.},
{110., 4427.}, {115., 3820.}, {120., 3307.}, {125., 2872.}];
```

```
In[30]:= ParameterCITable /. NonlinearRegress[
mgh10,  $\beta_1 e^{\frac{\beta_2}{x + \beta_3}}$ , x, {{\beta_1, 2}, {\beta_2, 400000}, {\beta_3, 25000}}},
RegressionReport → {ParameterCITable}, MaxIterations → 300]
```

Out[30]//TableForm=

	Estimate	Asymptotic SE	CI
β_1	0.00560964	0.000156879	{0.00527072, 0.00594855}
β_2	6181.35	23.309	{6130.99, 6231.7}
β_3	345.224	0.784861	{343.528, 346.919}

```
In[31]:= ParameterCITable /.
NonlinearRegress[mgh10,  $\beta_1 e^{\frac{\beta_2}{x + \beta_3}}$ , x, {{\beta_1, .02}, {\beta_2, 4000}, {\beta_3, 250}}},
RegressionReport → {ParameterCITable}, MaxIterations → 300]
```

Out[31]//TableForm=

	Estimate	Asymptotic SE	CI
β_1	0.00560964	0.000156879	{0.00527072, 0.00594855}
β_2	6181.35	23.309	{6130.99, 6231.7}
β_3	345.224	0.784861	{343.528, 346.919}

□ Bennett5

For this example, convergence in 5.1 can be obtained with half the iterations used in 5.0. From the first set of starting values, correct results are obtained with `MaxIterations`→750. From the second set of starting values, setting `MaxIterations`→200 is sufficient to obtain the benchmark results.

```
In[32]:= bennett5 = {{7.447168, -34.834702}, {8.102586, -34.3932},
  {8.452547, -34.152901}, {8.711278, -33.979099},
  {8.916774, -33.845901}, {9.087155, -33.732899}, {9.23259, -33.640301},
  {9.359535, -33.5592}, {9.472166, -33.486801}, {9.573384, -33.4231},
  {9.665293, -33.365101}, {9.749461, -33.313}, {9.827092, -33.260899},
  {9.899128, -33.2174}, {9.966321, -33.176899}, {10.02928, -33.139198},
  {10.08851, -33.101601}, {10.14443, -33.066799}, {10.19738, -33.035},
  {10.24767, -33.003101}, {10.29556, -32.971298}, {10.34125, -32.942299},
  {10.38495, -32.916302}, {10.42682, -32.890202}, {10.467, -32.864101},
  {10.50564, -32.841}, {10.54283, -32.817799}, {10.57869, -32.797501},
  {10.61331, -32.7743}, {10.64678, -32.757}, {10.67915, -32.733799},
  {10.71052, -32.7164}, {10.74092, -32.6991}, {10.77044, -32.678799},
  {10.7991, -32.6614}, {10.82697, -32.644001}, {10.85408, -32.626701},
  {10.88047, -32.612202}, {10.90619, -32.597698}, {10.93126, -32.583199},
  {10.95572, -32.568699}, {10.97959, -32.554298}, {11.00291, -32.539799},
  {11.0257, -32.525299}, {11.04798, -32.510799}, {11.06977, -32.499199},
  {11.0911, -32.487598}, {11.11198, -32.473202}, {11.13244, -32.461601},
  {11.15248, -32.435501}, {11.17213, -32.435501}, {11.19141, -32.4268},
  {11.21031, -32.4123}, {11.22887, -32.400799}, {11.24709, -32.392101},
  {11.26498, -32.380501}, {11.28256, -32.366001}, {11.29984, -32.3573},
  {11.31682, -32.348598}, {11.33352, -32.339901}, {11.34994, -32.3284},
  {11.3661, -32.319698}, {11.382, -32.311001}, {11.39766, -32.2994},
  {11.41307, -32.290699}, {11.42824, -32.282001}, {11.4432, -32.2733},
  {11.45793, -32.264599}, {11.47244, -32.256001}, {11.48675, -32.247299},
  {11.50086, -32.238602}, {11.51477, -32.2299}, {11.52849, -32.224098},
  {11.54202, -32.215401}, {11.55538, -32.2038}, {11.56855, -32.198002},
  {11.58156, -32.1894}, {11.59442, -32.183601}, {11.607121, -32.1749},
  {11.61964, -32.169102}, {11.632, -32.1633}, {11.64421, -32.154598},
  {11.65628, -32.145901}, {11.6682, -32.140099}, {11.67998, -32.131401},
  {11.69162, -32.125599}, {11.70313, -32.119801}, {11.71451, -32.111198},
  {11.72576, -32.1054}, {11.73688, -32.096699}, {11.74789, -32.0909},
  {11.75878, -32.088001}, {11.76955, -32.0793}, {11.7802, -32.073502},
  {11.79073, -32.067699}, {11.80116, -32.061901}, {11.81148, -32.056099},
  {11.8217, -32.050301}, {11.83181, -32.044498}, {11.84182, -32.038799},
  {11.85173, -32.033001}, {11.86155, -32.027199}, {11.87127, -32.0243},
  {11.88089, -32.018501}, {11.89042, -32.012699}, {11.89987, -32.004002},
  {11.90922, -32.001099}, {11.91849, -31.9953}, {11.92768, -31.9895},
  {11.93678, -31.9837}, {11.94579, -31.9779}, {11.95473, -31.972099},
  {11.96359, -31.969299}, {11.97237, -31.963501}, {11.98107, -31.957701},
  {11.9897, -31.9519}, {11.99826, -31.9461}, {12.00674, -31.9403},
  {12.01515, -31.937401}, {12.02349, -31.931601}, {12.03176, -31.9258},
  {12.03997, -31.922899}, {12.0481, -31.917101}, {12.05617, -31.911301},
  {12.06418, -31.9084}, {12.07212, -31.902599}, {12.08001, -31.8969},
  {12.08782, -31.893999}, {12.09558, -31.888201}, {12.10328, -31.8853},
```

```

{12.11092, -31.882401}, {12.1185, -31.8766}, {12.12603, -31.873699},
{12.1335, -31.867901}, {12.14091, -31.862101}, {12.14827, -31.8592},
{12.15557, -31.8563}, {12.16283, -31.8505}, {12.17003, -31.8447},
{12.17717, -31.841801}, {12.18427, -31.8389}, {12.19132, -31.833099},
{12.19832, -31.8302}, {12.20527, -31.827299}, {12.21217, -31.8216},
{12.21903, -31.818701}, {12.22584, -31.812901}, {12.2326, -31.809999},
{12.23932, -31.8071}, {12.24599, -31.8013}, {12.25262, -31.798401},
{12.2592, -31.7955}, {12.26575, -31.7897}, {12.27224, -31.7868}};

In[33]:= ParameterCITable /. NonlinearRegress[bennett5,
       $\beta_1 (x + \beta_2)^{-\frac{1}{\beta_3}}$ , x, {{ $\beta_1$ , -2000}, { $\beta_2$ , 50}, { $\beta_3$ , .8}},
      RegressionReport -> {ParameterCITable}, MaxIterations -> 750]

Out[33]//TableForm=

```

	Estimate	Asymptotic SE	CI
β_1	-2523.51	297.152	{-3110.62, -1936.39}
β_2	46.7366	1.24489	{44.2769, 49.1962}
β_3	0.932185	0.0202723	{0.892131, 0.972239}

```

In[34]:= ParameterCITable /. NonlinearRegress[bennett5,
       $\beta_1 (x + \beta_2)^{-\frac{1}{\beta_3}}$ , x, {{ $\beta_1$ , -1500}, { $\beta_2$ , 45}, { $\beta_3$ , .85}},
      RegressionReport -> {ParameterCITable}, MaxIterations -> 200]

Out[34]//TableForm=

```

	Estimate	Asymptotic SE	CI
β_1	-2523.51	297.152	{-3110.62, -1936.39}
β_2	46.7366	1.24489	{44.2769, 49.1962}
β_3	0.932185	0.0202723	{0.892131, 0.972239}

■ Acknowledgments

The research on which this article is based has been supported by the Maryland Agricultural Experiment Station. I am also indebted to two anonymous referees, as well as Daniel Lichtblau, Bruce McCullough, Charles Renfro, and, above all, Colin Rose for helpful comments. The author assumes responsibility for remaining errors.

■ Endnotes

(1) www.nag.co.uk/stats/ae_soft.asp

(2) www.itl.nist.gov/div898/strd

(3) McCullough [48] has also looked at several popular econometrics packages, such as E-Views, LIMDEP, SHAZAM, and TSP. The results are generally comparable to those obtained for SAS.

■ References

- [1] B. D. McCullough and H. D. Vinod, "The Numerical Reliability of Econometric Software," *Journal of Economic Literature*, **37**, 1999 pp. 633–665.
- [2] B. D. McCullough and B. Wilson, "On the Accuracy of Statistical Procedures in Microsoft Excel 97," *Computational Statistics and Data Analysis*, **31**, 1999 pp. 27–37.
- [3] B. D. McCullough, "The Accuracy of Mathematica 4 as a Statistical Package," *Computational Statistics*, **15**, 2000 pp. 279–299.
- [4] H. D. Vinod, "Review of GAUSS for Windows, Including Its Numerical Accuracy," *Journal of Applied Econometrics*, **15**, 2000 pp. 211–220.
- [5] B. D. McCullough, "Assessing the Reliability of Statistical Software: Parts I and II," *The American Statistician*, **52**, 1998 pp. 358–366; **53**, 1999 pp. 149–159.
- [6] J. Norris, Calibration of Ozone Monitors, NIST, unpublished.
- [7] P. Pontius, Load Cell Calibration, NIST, unpublished.
- [8] K. Eberhardt, Nolnt1 and Nolnt2, NIST, unpublished.
- [9] A. Filippelli, Filip, NIST, unpublished.
- [10] J. W. Longley, "An Appraisal of Least Squares Programs for the Electronic Computer from the Viewpoint of the User," *Journal of the American Statistical Association*, **62**, 1967 pp. 819–841.
- [11] R. H. Wampler, "A Report of the Accuracy of Some Widely-Used Least Squares Computer Programs," *Journal of the American Statistical Association*, **65**, 1970 pp. 549–565.
- [12] D. Misra, Dental Research Monomolecular Adsorption Study, NIST, unpublished, 1978.
- [13] D. Chwirut, Ultrasonic Reference Block Study, NIST, unpublished, 1979.
- [14] C. Lanczos, *Applied Analysis*, Englewood Cliffs, NJ: Prentice Hall, 1956 pp. 272–280.
- [15] B. Rust, Gauss 1, 2, and 3, NIST, unpublished, 1996.
- [16] C. Daniel and F. S. Wood, *Fitting Equations to Data*, 2nd ed., New York: John Wiley & Sons, 1980 pp. 428–431.
- [17] R. Kirby, Scanning Electron Microscope Line Width Standards, NIST, unpublished, 1979.
- [18] T. Hahn, Copper Thermal Expansion Study, NIST, unpublished, 1979.
- [19] W. Nelson, "Analysis of Performance-Degradation Data," *IEEE Transactions on Reliability*, **2**, R-30 (2), 1981 pp. 149–155.
- [20] M. R. Osborne, "Some Aspects of Nonlinear Least Squares Calculations," *Numerical Methods for Nonlinear Optimization* (F. A. Lootsma, ed.), New York: Academic Press, 1972 pp. 171–189.
- [21] L. Roszman, Quantum Defects for Sulfur I Atom, NIST, unpublished, 1979.
- [22] D. Kahaner, C. Moler, and S. Nash, *Numerical Methods and Software*, Englewood Cliffs, NJ: Prentice Hall, 1989 pp. 441–445.
- [23] J. S. Kowalik and M. R. Osborne, *Methods for Unconstrained Optimization Problems*, New York: Elsevier North-Holland, 1978.

- [24] R. Thurber, Semiconductor Electron Mobility Modeling, NIST, unpublished, 1979.
- [25] G. P. Box, W. G. Hunter, and J. S. Hunter, *Statistics for Experimenters*, New York: John Wiley & Sons, 1978 pp. 483–487.
- [26] D. A. Ratkowsky, *Nonlinear Regression Modelling: A Unified Practical Approach*, New York: Marcel Dekker, 1983 pp. 61–62 and 88.
- [27] R. R. Meyer, "Theoretical and Computational Aspects of Nonlinear Regression," *Nonlinear Programming* (J. B. Rosen, O. L. Mangasarian, and K. Ritter, eds.), New York: Academic Press, 1970 pp. 465–486.
- [28] K. Eckerle, Circular Interference Transmittance Study, NIST, unpublished, 1979.
- [29] L. Bennett, L. Swartzendruber, and H. Brown, Bennett5, NIST, unpublished, 1994.
- [30] K. L. Hiebert, "An Evaluation of Mathematical Software that Solves Nonlinear Least Squares Problems: Part I," *ACM Transactions on Mathematical Software*, **7**, 1981 pp. 1–16.
- [31] S. Wolfram, *The Mathematica Book*, 5th ed., Champaign, Oxford: Wolfram Media/Cambridge University Press, 2003.
- [32] R. Davidson and J. G. MacKinnon, "Nonlinear Regression Models and Nonlinear Least Squares," *Estimation and Inference in Econometrics*, New York: Oxford University Press, 1993 pp. 41–65.
- [33] W. H. Greene, "Nonlinear Regression Models," *Econometric Analysis*, 4th ed., Englewood Cliffs, NJ: Prentice Hall, 2000 pp. 416–454.
- [34] W. Härdle, *Applied Nonparametric Regression*, New York: Cambridge University Press, 1990.
- [35] A. R. Pagan and A. Ullah, *Nonparametric Econometrics*, New York: Cambridge University Press, 1998.
- [36] A. R. Gallant, *Nonlinear Statistical Models*, New York: John Wiley & Sons, 1987.
- [37] D. M. Bates and D. G. Watts, *Nonlinear Regression Analysis and Its Applications*, New York, John Wiley & Sons, 1988.
- [38] G. A. F. Seber and C. J. Wild, *Nonlinear Regression*, New York: John Wiley & Sons, 1989.
- [39] A. R. Gallant, "Nonlinear Regression," *The American Statistician*, **29**, 1975 pp. 73–81.
- [40] H. O. Hartley, "The Modified Gauss-Newton Method for the Fitting of Nonlinear Regression Functions by Least Squares," *Technometrics*, **3**, 1961 pp. 269–280.
- [41] H. O. Hartley and A. Booker, "Non-linear Least Squares Estimation," *The Annals of Mathematical Statistics*, **36**, 1965 pp. 638–650.
- [42] D. W. Marquardt, "An Algorithm for Least-Squares Estimation of Nonlinear Parameters," *Journal of the Society for Industrial and Applied Mathematics*, **11**, 1963 pp. 431–441.
- [43] K. Levenberg, "A Method for the Solution of Certain Nonlinear Problems in Least Squares," *Quarterly of Applied Mathematics*, **2**, 1944 pp. 164–168.
- [44] K. L. Hiebert, "An Evaluation of Mathematical Software That Solves Nonlinear Least Squares Problems: Part II," *ACM Transactions on Mathematical Software*, **8**, 1982 pp. 5–20.
- [45] R. I. Jennrich, "Asymptotic Properties of Nonlinear Least Squares Estimators," *Annals of Mathematical Statistics*, **40**, 1969 pp. 633–643.

- [46] J. J. More, B. S. Garbow, and K. E. Hillstom, "Testing Unconstrained Optimization Software," *ACM Transactions on Mathematical Software*, **7**, 1981 pp. 17–41.
- [47] R. D. Skeel and J. B. Keiper, *Elementary Numerical Computing with Mathematica*, New York: McGraw-Hill, 1993.
- [48] B. D. McCullough, "The Reliability of Econometric Software: E-Views, LIMDEP, SHAZAM, and TSP," *Journal of Applied Econometrics*, **14**, 1999 pp. 191–202.

■ Additional Material

AppendicesAccuracy.nb

Available at www.mathematica-journal.com/issue/v9i4/download.

About the Author

Marc Nerlove is a professor of economics at the University of Maryland, College Park, MD. He received his B.A. in mathematics from the University of Chicago and his M.A. and Ph.D. degrees in economics from Johns Hopkins University. His research interests include environmental and resource economics, likelihood inference in econometrics, and agricultural development in the context of two-sector economic growth.

Marc Nerlove

Department of Agricultural and Resource Economics
University of Maryland
College Park, MD 20742
mnerlove@arec.umd.edu
www.arec.umd.edu/mnerlove/mnerlove.htm

Skew Densities and Ensemble Inference for Financial Economics

H. D. Vinod

Financial economics includes the study of the density of ‘excess returns,’ having skewness, fat tails, and excess kurtosis. We illustrate it using monthly data on a mutual fund. The nonnormality of $f(x)$ impacts measurement of potential losses by ‘value at risk’ (VaR). Since finance literature has mostly ignored Azzalini’s [1] skew normal (SN) density, our aim is to illustrate its implementation using *Mathematica* and *mathStatica* [2]. We use a location-scale version of the SN distribution, create its likelihood function, and use maximum likelihood (ML) to estimate its parameters. We illustrate it with data for a mutual fund and report the ML estimates, quantiles of $f(x)$, and the VaR.

■ 1. Introduction

Excess return from an asset (e.g., a mutual fund) is defined as the return over and above the risk-free rate (e.g., interest rate earned on the three-month U.S. Treasury Bills). Recent literature in financial economics recognizes that the density of excess returns $f(x)$ is often nonnormal due to skewness, kurtosis, and fat tails. While some extensions of the normal called log-normal and inverse Gaussian have been used, Azzalini’s [1] SN density appears to have been ignored. The aim of this article is to introduce *mathStatica* implementation from [3]. This software also enables easy estimation of several other nonnormal $f(x)$ for myriad potential uses in finance. We discuss ML estimators for the parameters of the SN density.

Wall Street investors, bankers, and government regulators often want a dollar figure on the potential loss in a worst-case scenario. VaR is one such measure developed by statisticians at RiskMetrics Group, a private company in New York. VaR is often described as based on historical volatility and tells the investor what might happen if some unusual event made the asset more volatile than normal. Although actual measurements differ among analysts, VaR is often obtained from a low (e.g., 1%) quantile of a parametric or nonparametric $f(x)$. Since mutual fund investors have seen three consecutive down years, both investors and regulators are concerned about fund VaR values. Morgenstern [4] notes that

risky funds in the aggressive growth stock category have recently experienced \$5.41 billion in redemptions compared to May 2002, whereas fund managers have increased assets in the aggressive category from \$430 billion to \$1.3 trillion. Our point is that VaR is important in finance, and we propose simple tools for calculating it with the help of *mathStatistica*.

Section 2 considers the Alliance All-Asia Investment Advisors Fund mutual fund, with the ticker symbol AAAYX, for the period of $T = 132$ months from January 1987 to December 1997 from [5]. Descriptive statistics reveal that the density $f(x)$ is negatively skewed. Section 3 provides a generalised SN model adjusted for location and scale. Section 4 discusses ML estimation of the parameters of the generalised SN density. Section 5 briefly discusses VaR.

In future work, we expect to develop *mathStatistica* software for a new maximum entropy algorithm (ME-alg) for the time series inference suggested in [6] for inference regarding the VaR. Classical time series inference methods treat the observed time series x_t as one random realization from an unobservable ensemble Ω . These were developed in the 1930s before computers and before Shannon information and entropy ideas were developed. The ME-alg suggests a computer intensive construction of an approximate Ω built up from elements Ω_j for $j = 1, 2, \dots, J$ with J a large number. Section 6 contains our conclusions.

We begin by loading the *mathStatistica* package.

```
In[1]:= << mathStatistica.m
```

■ 2. The Empirical Data

The data for AAAYX, *ydata*, is loaded as follows.

```
In[2]:= ydata = {10.724', 1.737', 2.123', 0.031', -0.483', 3.913', 4.284',
  2.275', -2.462', -21.147', -6.062', 6.318', 3.221', 2.623',
  -3.985', 0.792', 1.644', 4.042', -1.892', -3.922', 2.683', 0.673',
  -1.167', 1.11', 5.084', -3.543', 0.24', 3.783', 3.165', -2.281',
  5.836', 2.028', -1.974', -3.262', 2.253', 2.256', -6.66', 2.338',
  0.935', -4.164', 6.84', -2.66', -0.538', -9.629', -4.608',
  -0.705', 5.103', 2.153', 3.574', 4.487', -0.388', -0.568', 3.93',
  -4.971', 2.93', 1.948', -0.134', 1.391', -4.869', 8.073', -1.325',
  1.167', -1.343', 3.987', 0.887', -1.302', 3.231', -2.667',
  1.573', -0.45', 2.824', 2.04', -0.145', 2.051', 2.756', -0.746',
  0.741', 0.144', 0.644', 3.246', -0.447', 0.444', -1.266', 2.146',
  2.55', -2.184', -4.595', 0.572', 1.743', -1.852', 2.738', 4.213',
  -1.996', 0.973', -4.072', 2.03', 0.505', 4.011', 2.116', 1.517',
  3.122', 1.038', 1.84', 0.651', 2.955', -2.352', 5.648', 1.379',
  2.483', 0.585', 0.976', 0.275', 0.772', -1.428', -4.54', 2.863',
  4.484', 2.774', 6.676', -2.028', 3.673', 1.067', -2.84', 3.661',
  3.971', 3.871', 7.067', -5.332', 4.383', -4.93', 3.762', 0.058'};
```

The descriptive statistics for our data are given by using `Statistics`DescriptiveStatistics``.

```

In[3]:= << Statistics`DescriptiveStatistics`
In[4]:= LocationReport[ydata]
Out[4]= {Mean → 0.738705, HarmonicMean → 1.95493, Median → 1.0885}

In[5]:= DispersionReport[ydata]
Out[5]= {Variance → 14.1743, StandardDeviation → 3.76488,
  SampleRange → 31.871, MeanDeviation → 2.68482,
  MedianDeviation → 2.055, QuartileDeviation → 2.11325}

In[6]:= ShapeReport[ydata]
Out[6]= {Skewness → -1.58578,
  QuartileSkewness → -0.122678, KurtosisExcess → 7.69132}

```

These reports show that the usual normal distribution is not suitable for our data and that the negative skewness requires extra care in interpreting the results. To illustrate graphically, *mathStatica* can be used for nonparametric kernel density estimation in two steps. First, we specify the kernel as the Gaussian kernel. The second step is to choose the bandwidth denoted here by c .

```

In[7]:= Gauss =  $\frac{e^{-\frac{y^2}{2}}}{\sqrt{2\pi}}$ ;
          domain[Gauss] = {y, -∞, ∞};

In[9]:= c = Bandwidth[ydata, Gauss, Method → SheatherJones]
Out[9]= 1.104

```

Since the kernel K and c have now been specified, we can plot Figure 1 as our smoothed nonparametric kernel density estimate using the `NP KDEPlot[data, K, c]` function.

```

In[10]:= NP KDEPlot[ydata, Gauss, c];

```

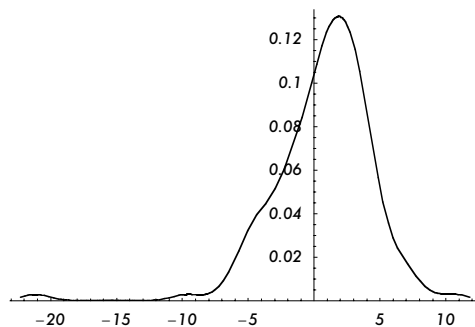


Figure 1.

The preceding discussion and plot illustrate a common situation: normality is often not suitable for many financial time series including ours. *mathStatica* can

effectively implement a wide range of nonnormal densities for applications in finance.

■ 3. Model: A Generalised Azzalini Skew Normal Distribution

Next, we illustrate the power of *mathStatica* and *Mathematica* in financial economics by implementing a generalised version of Azzalini's SN density in the context of our example. Azzalini [1] proved that if a random variable X has a pdf $f(x)$ which is symmetric about 0, and cdf $F(x)$, then $g(x) = 2 f(x) F(\lambda x)$ is also a pdf, for parameter $\lambda \in \mathbb{R}$. If $X \sim N(0, 1)$, then $g(x)$ is Azzalini's SN density. Unfortunately, the latter does not contain sufficient flexibility for our financial application which will require both a scale parameter (for large variance) and a location parameter.

To start our generalisation, we first introduce a scale parameter $w > 0 \dots$ hence, let $X \sim N(0, w^2)$ with pdf $f(x)$.

$$\begin{aligned} \text{In[11]:= } f &= \frac{1}{w \sqrt{2 \pi}} \text{Exp}\left[-\frac{x^2}{2 w^2}\right]; \\ \text{domain}[f] &= \{x, -\infty, \infty\} \&\& \{w > 0\}; \end{aligned}$$

The cdf evaluated at $F(\lambda x)$ is:

$$\begin{aligned} \text{In[13]:= } F &= \text{Prob}[\lambda x, f] \\ \text{Out[13]= } &\frac{1}{2} \left(1 + \text{Erf}\left[\frac{x \lambda}{\sqrt{2} w}\right]\right) \end{aligned}$$

Then, our generalised Azzalini skew $N(0, w^2)$ density, $g(x)$, is simply:

$$\begin{aligned} \text{In[14]:= } g &= 2 * f * F \\ \text{Out[14]= } &\frac{e^{-\frac{x^2}{2 w^2}} \left(1 + \text{Erf}\left[\frac{x \lambda}{\sqrt{2} w}\right]\right)}{\sqrt{2 \pi} w} \end{aligned}$$

with domain of support:

$$\text{In[15]:= } \text{domain}[g] = \{x, -\infty, \infty\} \&\& \{\lambda \in \text{Reals}, w > 0\};$$

Next, we introduce a location parameter ξ into the model by transforming $X \rightarrow Y$ such that $Y = X + \xi$. This is facilitated by *mathStatica*'s `Transform[ ]` function to yield pdf $h(y)$.

$$\begin{aligned} \text{In[16]:= } h &= \text{Transform}[y == x + \xi, g] \\ \text{Out[16]= } &\frac{e^{-\frac{(y-\xi)^2}{2 w^2}} \left(1 + \text{Erf}\left[\frac{\lambda (y-\xi)}{\sqrt{2} w}\right]\right)}{\sqrt{2 \pi} w} \end{aligned}$$

with domain of support:

$$\text{In[17]:= } \text{domain}[h] = \{y, -\infty, \infty\} \&\& \{\lambda \in \text{Reals}, \xi \in \text{Reals}, w > 0\};$$

Figure 2 plots our generalised location-scale SN density for three values of the key parameter λ to verify that its negative values lead to negative skewness in a graph. Note that if a mutual fund has $\lambda = -9$ (other things being equal), it is mostly losing money. In the sequel, we shall estimate λ by maximizing the likelihood function.

```
In[18]:= PlotDensity[h /. {ξ → 0, w → 1, λ → {0, -2, -9}}, {y, -3, 3}];
```

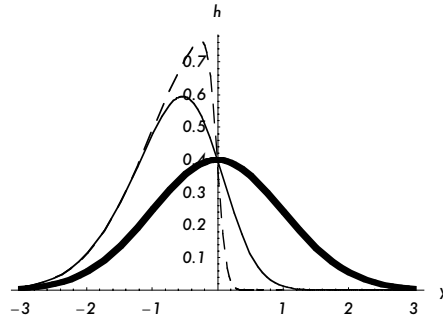


Figure 2.

■ 4. Maximum Likelihood Estimation

Next, we activate *mathStatica*'s `SuperLog` function for subsequent calculation of the log likelihood for $b(y)$.

```
In[19]:= SuperLog[On]
```

SuperLog is now On.

Now we are ready to define $\log L\theta$, the log of the likelihood function, from a product of $b(y)$ densities over n observations.

```
In[20]:= logLθ = Log[∏i=1n (h /. y → yi) ]
```

$$\begin{aligned} \text{Out[20]} = & -\frac{1}{2w^2} \left(n\xi^2 + nw^2 \text{Log}[2] + nw^2 \text{Log}[\pi] + 2nw^2 \text{Log}[w] - \right. \\ & \left. 2w^2 \sum_{i=1}^n \text{Log}\left[1 + \text{Erf}\left[\frac{\lambda(-\xi + y_i)}{\sqrt{2}w}\right]\right] - 2\xi \sum_{i=1}^n y_i + \sum_{i=1}^n y_i^2 \right) \end{aligned}$$

The ML estimates of the three parameters ξ , λ , and w require differentiation of the $\log L\theta$, leading to three so-called score functions.

In[21]:= **score = Grad[logLθ, {ξ, λ, w}]**

$$\text{Out[21]} = \left\{ -\frac{2n\xi - 2w^2 \sum_{i=1}^n -\frac{e^{-\frac{\lambda^2(-\xi+y_i)^2}{2w^2}} \sqrt{\frac{2}{\pi}} \lambda}{w(1+\text{Erf}[\frac{\lambda(-\xi+y_i)}{\sqrt{2}w}])} - 2 \sum_{i=1}^n y_i}{2w^2}, \right. \\ \sum_{i=1}^n \frac{e^{-\frac{\lambda^2(-\xi+y_i)^2}{2w^2}} \sqrt{\frac{2}{\pi}} (-\xi+y_i)}{w(1+\text{Erf}[\frac{\lambda(-\xi+y_i)}{\sqrt{2}w}])}, \\ \frac{1}{w^3} \left(n\xi^2 + n w^2 \text{Log}[2] + n w^2 \text{Log}[\pi] + 2n w^2 \text{Log}[w] - \right. \\ \left. 2w^2 \sum_{i=1}^n \text{Log}\left[1 + \text{Erf}\left[\frac{\lambda(-\xi+y_i)}{\sqrt{2}w}\right]\right] - 2\xi \sum_{i=1}^n y_i + \sum_{i=1}^n y_i^2 \right) - \\ \frac{1}{2w^2} \left(2nw + 2nw \text{Log}[2] + 2nw \text{Log}[\pi] + 4nw \text{Log}[w] - \right. \\ \left. 4w \sum_{i=1}^n \text{Log}\left[1 + \text{Erf}\left[\frac{\lambda(-\xi+y_i)}{\sqrt{2}w}\right]\right] - \right. \\ \left. 2w^2 \sum_{i=1}^n -\frac{e^{-\frac{\lambda^2(-\xi+y_i)^2}{2w^2}} \sqrt{\frac{2}{\pi}} \lambda (-\xi+y_i)}{w^2(1+\text{Erf}[\frac{\lambda(-\xi+y_i)}{\sqrt{2}w}])} \right) \Bigg\}$$

These equations are too complicated to analytically obtain the ML estimators by setting the scores equal to zero. We have to use numerical solution methods on our data.

The observed log likelihood is obtained by simply inserting the data values in $\log L\theta$.

In[22]:= **obslogLθ = logLθ /. {n → Length[ydata], y_i → ydata[[i]]};**

We then use the FindMaximum function to numerically maximize the observed log likelihood, as follows:

In[23]:= **sol = FindMaximum[obslogLθ, {{ξ, 0.1}, {λ, -0.2}, {w, 2}}]**

FindMaximum::lstol :

The line search decreased the step size to within tolerance specified by AccuracyGoal and PrecisionGoal but was unable to find a sufficient increase in the function. You may need more than MachinePrecision digits of working precision to meet these tolerances. More . . .

Out[23]= {-361.793, {ξ → 0.738673, λ → 0.0000106647, w → 3.75059}}

In the preceding output, the first component of the solution is the value of the maximized likelihood. The second component is sol[[2]] and represents the ML solution for the three parameters. *Mathematica*'s warning message suggests that

we should try other methods and starting points. We will try another search, now using `Method → Gradient` and slightly different starting points.

```
In[24]:= sol1 = FindMaximum[obslogLθ, {ξ, 0.78},
      {λ, -0.014}, {w, 3.75}, Method → Gradient]
Out[24]= {-361.342, {ξ → 2.40441, λ → -0.55144, w → 3.779}}
```

The new output in `sol1` is already an improvement. But is it the maximum? By the first-order conditions, the gradients of the observed $\log L\theta$ evaluated at `sol1[[2]]` should be zero.

```
In[25]:= grd = Grad[obslogLθ, {ξ, λ, w}];
      grd /. sol1[[2]]
Out[26]= {-2.18193, -6.30854, 5.34271}
```

These values of the gradient clearly are not close enough to zero. Similarly, the second-order conditions for a maximum require that the matrix H of second-order partials be negative definite. `Hessian` calculates this matrix, and `Eigenvalues` calculates its eigenvalues as follows:

```
In[27]:= hess = Hessian[obslogLθ, {ξ, λ, w}]; Eigenvalues[hess /. sol1[[2]]]
Out[27]= {-61.5201, -25.7395, 0.293474}
```

Unfortunately, the last eigenvalue, although small, does not ensure negative definiteness. The issue is whether the observed log likelihood is concave near the solution. McCullough and Vinod [7, 8] recommend searching near all ML solutions to be certain. Fortunately, *Mathematica* makes these searches relatively easy to implement by specifying an alternative method. Hence we try the `Newton` estimation method.

```
In[28]:= sol2 = FindMaximum[obslogLθ,
      {ξ, 0.78}, {λ, -0.014}, {w, 3.75}, Method → Newton]
Out[28]= {-354.229, {ξ → 4.54772, λ → -2.30187, w → 5.34561}}
```

Is this solution (denoted by `sol2`) superior to the first one? The maximized value of the log likelihood has increased from -361.342 to -354.229 . So the solution has clearly improved. The solution for the location parameter ξ is not comparable to the sample mean 0.7387 for our mutual fund, because of the scale change by w . The comparable value is obtained as:

```
In[29]:= ξ2 = ξ / w;
      ξ2 /. sol2[[2]]
Out[30]= 0.85074
```

To determine if `sol2` is good enough we study its properties as before.

```
In[31]:= grd /. sol2[[2]]
Out[31]= {-1.89938 × 10-8, -2.48643 × 10-8, 3.99092 × 10-9}
In[32]:= Eigenvalues[hess /. sol2[[2]]]
Out[32]= {-16.0411, -12.9347, -1.70855}
```

This solution has the desirable property that the gradients are very close to zero. The shape of the log likelihood near this second ML solution is concave, since the Hessian is negative definite. For a graphical assessment of the goodness-of-fit of the model to the data, we plot the data (solid line) and the fitted location-scale SN density $b(y)$ (dashed line) side by side in Figure 3.

```
In[33]:= FrequencyPlot[ydata, h /. sol2[[2]]];
```

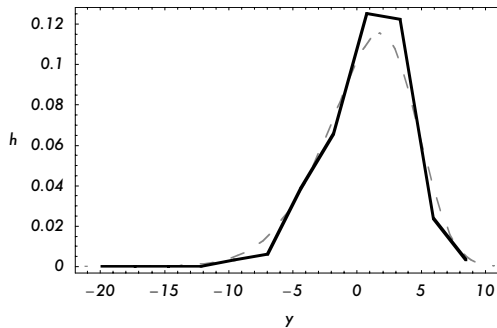


Figure 3.

Since this fit is visually close, we will accept `sol2` as our solution. The next step is to consider the statistical inference for the Newton solution of the three parameters. It is well known that the negative of the inverse of the Hessian matrix evaluated at the solution provides the asymptotic variance-covariance matrix of the ML parameter estimates.

```
In[34]:= asycov = -Inverse[hess /. sol2[[2]]]
```

```
Out[34]:= 
$$\begin{pmatrix} 0.216946 & -0.180039 & 0.154585 \\ -0.180039 & 0.289606 & -0.174896 \\ 0.154585 & -0.174896 & 0.218391 \end{pmatrix}$$

```

The standard errors of the ML estimates are given by the diagonals of the preceding matrix. Hence we write student's *t* statistics as:

```
In[35]:= 
$$\frac{\{\xi, \lambda, w\} /. sol2[[2]]}{\sqrt{\text{Table}[asy cov[[i, i]], \{i, 3\}]}}$$

```

```
Out[35]:= {9.76378, -4.27738, 11.4388}
```

Note that the *t* statistics are scale-free and suggest statistical significance of the three parameters at conventional confidence levels. We conclude that this mutual fund yields statistically significant positive return even after allowing for negative skewness in the data. The ML estimate of the skewness parameter, λ is -2.30187 , is also of interest in financial economics in the following sense. Note that except for location and scale change, the density for our fund is close to the dotted line in Figure 3. If there are two funds with identical mean and variance but distinct skewnesses, then the investor would be better off choosing the fund with a larger (more positive) skewness.

The asymptotic inference may not be satisfactory in relatively small samples in finance. In future work we expect to develop a new maximum entropy algorithm (ME-alg) for the time series inference suggested in [6] for such inference.

■ 5. Value at Risk

The VaR is a popular measure of potential loss used by investors, bankers, and regulators. Let $\Delta P(\tau) = P(t + \tau) - P(t)$ be the change in the value of a portfolio at time t for horizon τ and let α (< 0.5) denote a probability. The VaR is defined by the probability statement:

$$\Pr[\Delta P(\tau) < -\text{VaR}] = 1 - \alpha, \quad (1)$$

where the negative VaR is designed to measure losses in positive dollars. Intuitively, VaR measures a worst-case scenario loss associated with 'long' positions (buying side). For example, consider an investor with a time horizon τ of one year buying \$100,000 worth of mutual fund shares, $P(t) = 100000$. Now assume that the fund could lose 25% or more of its value in a year with probability $\alpha = 0.01$. Then $\Pr[\Delta P(\tau) < 25000] = 0.99$, implying that $\text{VaR} = \$25,000$ is an upper bound on the loss. Note that we can compute the VaR for any density whose cdf is known, at least numerically.

```
In[36]:= Quantile[ydata, 0.01]
```

```
Out[36]= -9.629
```

For our original data in terms of returns, the low return of -9.629% suggests $\text{VaR} = \$9629$. It is customary not to use the estimate based on the empirical cdf, since the investor is concerned with worst-case scenarios, beyond what was experienced. One way to do this has been to use the lower one percentile of $N(0, 1)$, namely -2.33 .

```
In[37]:= Mean[ydata] - 2.33 * Sqrt[Variance[ydata]]
```

```
Out[37]= -8.03347
```

This suggests that $\text{VaR} = \$8033$. It is somewhat surprising that the VaR from $N(0, 1)$ is less than \$9629, from the quantile of the original data. We have not yet computed the VaR from $h(y)$. First, let us check its cdf as:

```
In[38]:= Prob[y, h]
```

$$\text{Out[38]} = \int_{-\infty}^y \frac{e^{-\frac{(y-\xi)^2}{2w^2}} (1 + \text{Erf}[\frac{\lambda(y-\xi)}{\sqrt{2}w}])}{\sqrt{2\pi}w} dy$$

Since inverting this analytically is not feasible, we use numerical methods as follows. A numerical version of the density is:

```
In[39]:= h2 = h /. sol2[[2]]
```

```
Out[39]= 0.0746299 e^{-0.0174975 (-4.54772+y)^2} (1 + Erf[-0.304487 (-4.54772+y)])
```

The numerical cdf is obtained by integrating $b_2(y)$ as follows:

```
In[40]:= NProb[r_?NumericQ] := NIntegrate[h2, {y, -∞, r}]
```

Next, we ask *Mathematica* to search for the value of Y at which the numerical cdf is equal to 0.01.

```
In[41]:= FindRoot[NProb[r] == 0.01, {r, -15, -5}]
```

```
Out[41]= {r → -9.22166}
```

Hence the VaR estimated from the location-scale SN density is \$9222, rounded to the nearest dollar. Vinod and Morey [9] note that financial economists usually ignore the estimation risk. Since this is also true in calculated VaR estimates, there is a need to develop inference methods for VaR. In future work, using tools described in Section 2.6 of [3] we seek to compute pseudo-random number generation from our $b(y)$ and also from the kernel density plotted in Figure 1. The large number ($J = 999$, say) of realizations of VaR can then be ordered from the smallest to the largest and denoted by $\text{VaR}(j)$ with $j = 1, 2, \dots, J$. An obvious possibility is to choose the $\text{VaR}(10)$ as the estimate of VaR which incorporates both estimation risk and investment risk.

■ 6. Conclusion

We provide descriptive statistics and graphics which show that monthly excess returns data for a mutual fund reveals that their distribution $f(x)$ is nonnormal. Since the SN distribution has been generally ignored by financial economists, we consider its potential in describing $f(x)$. Using *mathStatistica*, [2], we derive the score function for the parameters ξ , λ , and w of a location-scale SN density and their ML estimators. We note that *mathStatistica* and *Mathematica* provide tools for numerous potential uses in finance, including ML estimation of parameters of several nonnormal $f(x)$. We show that these densities can be further used to compute low percentiles (1%) to help compute the VaR, which is gaining considerable use by Wall Street practitioners and the financial media [4].

■ References

- [1] A. Azzalini, "A Class of Distributions Which Includes the Normal Ones," *Scandinavian Journal of Statistics*, **12**, 1985 pp. 171–178.
- [2] *mathStatistica*, Sydney, Australia: MATHSTATICA Pty. Ltd., (Feb 2005) www.mathStatistica.com.
- [3] C. Rose and M. D. Smith, *Mathematical Statistics with Mathematica*, New York: Springer-Verlag, 2002.
- [4] G. Morgenstern, "How Big Could the Next Wave Be? Check the Risk Level," Market Watch section, *New York Times*, Feb 1, 2003.
- [5] Morningstar, *On-Disk, Principia, Principia-Plus, and Principia-Pro Manuals* (1991–2000), Chicago: Morningstar Publications, 2000.

- [6] H. D. Vinod, "Maximum Entropy Constructive Ensembles for Time Series Analysis" (discussion paper, Department of Economics, Fordham University, Bronx, NY, Dec 22, 2002). Paper presented at the 35th Symposium on the Interface: Computing Science and Statistics (Salt Lake City, UT, Mar 14, 2003) is available on CD-ROM from the Interface Foundation of North America, Inc., P. O. Box 7460, Fairfax Station, VA 22039-7460. Also available at www.american.edu/cas/econ/faculty/golan/Papers/9_17vinod.doc.
- [7] B. D. McCullough and H. D. Vinod, "The Numerical Reliability of Econometric Software," *Journal of Economic Literature*, **37**(2), 1999 pp. 633–665.
- [8] B. D. McCullough and H. D. Vinod, "Verifying the Solution from a Nonlinear Solver: A Case Study," *The American Economic Review*, **93**(3), June 2003 pp. 873–892.
- [9] H. D. Vinod and M. R. Morey, "Estimation Risk in Morningstar Fund Ratings," *Journal of Investing*, **11**(4), 2002 pp. 67–75.

About the Author

Professor H. D. Vinod is a professor of economics and director of the Institute for Ethics and Economic Policy at Fordham University in New York. He received his Ph.D. in economics from Harvard University, has worked at Bell Laboratories for many years, and has been an expert witness in major antitrust trials. He has published over a hundred articles in top journals in econometrics and statistics, including *Econometrica*, *Journal of Econometrics*, *Journal of the American Statistical Association*, *Management Science*, *American Economic Review*, *Review of Economics and Statistics*, *International Economic Review*, *Journal of Economic Literature*, *Journal of Business*, and *Journal of Advertising Research*. He has edited special issues and refereed many journals. In 1996, he was named a fellow of the *Journal of Econometrics*. Maharashtra Foundation honored him for outstanding social service in 1998. Vinod coauthored a 1981 research monograph on regression methods and a 2005 Wiley monograph, "Preparing for the worst: Incorporating downside risk in stock market investing." He co-edited volume 11 of the *Handbook of Statistics* with C. R. Rao, the famous statistician. Vinod manages a worldwide anticorruption project described at www.fordham.edu/economics/vinod.

H. D. Vinod

Professor
Economics Department
Fordham University
Bronx, New York 10458
Vinod@fordham.edu

Neighbourhoods of Independence for Random Processes via Information Geometry

Khadiga Arwini
C. T. J. Dodson

In this article we consider the Freund family of distributions as a 4-manifold equipped with Fisher information as Riemannian metric and derive the induced α -geometry, that is, the α -Ricci curvature, the α -scalar curvature, and so forth. We show that the Freund manifold has a positive constant 0-scalar curvature, so geometrically it constitutes part of a sphere. We examine special cases as submanifolds and discuss their geometrical structures; via one submanifold we provide examples of neighbourhoods of the independent case for bivariate distributions having identical exponential marginals. Thus, since exponential distributions complement Poisson point processes, we obtain a means to discuss the neighbourhood of independence for random processes. Our approach using *Mathematica* handles the α -geometry calculations and graphics effectively and could be transferred to other distribution families.

■ 1. Differential Geometry of the Freund 4-Manifold F

In [1] we proved that every neighbourhood of an exponential distribution contains a neighbourhood of gamma distributions in the subspace topology of $\mathbb{R}^3$, using information geometry (see Amari et al. [2, 3]) and the affine immersion of Dodson and Matsuzoe [4]. As part of a study of the information geometry of gamma and bivariate gamma stochastic processes [5, 6, 7], we have calculated the geometry of the family of Freund bivariate mixture exponential density functions. The importance of this family lies in the fact that exponential distributions represent intervals between events for Poisson processes on the real line, and the Freund family provides models for bivariate processes with positive and negative covariance.

The Freund family of distributions becomes a Riemannian 4-manifold with the Fisher information metric, and we derive the induced α -geometry, that is, the α -Ricci curvature, the α -scalar curvature, and so forth. We also show that the Freund manifold has a positive constant 0-scalar curvature, so geometrically it constitutes part of a sphere.

□ 1.1. Freund Bivariate Mixture Exponential Distributions

This model was one of the first bivariate distributions to be obtained from reliability considerations. Freund [8] introduced a bivariate exponential mixture distribution arising from the following situation. Suppose that an instrument has two components, A and B , with lifetimes X and Y having density functions (when both components are in operation) $f_X(x) = \alpha_1 e^{-\alpha_1 x}$; $f_Y(y) = \alpha_3 e^{-\alpha_3 y}$, ($\alpha_1, \alpha_3 > 0$; $x, y > 0$). Then X and Y are dependent, in that a failure of either component changes the parameter of the life distribution of the other component. Thus, when A fails, the parameter for Y becomes α_4 ; when B fails, the parameter for X becomes α_2 . There is no other dependence. The joint density function of X and Y is

$$f(x, y) = \begin{cases} \alpha_1 \alpha_4 e^{-\alpha_4 y - (\alpha_1 + \alpha_3 - \alpha_4) x} & \text{for } 0 < x < y \\ \alpha_3 \alpha_2 e^{-\alpha_2 x - (\alpha_1 + \alpha_3 - \alpha_2) y} & \text{for } 0 < y < x, \end{cases} \quad (1)$$

where $\alpha_i > 0$ ($i = 1, 2, 3, 4$).

Define the functions f_1 and f_2 :

$$f_1 := f_1 = \alpha_1 \alpha_4 e^{-\alpha_4 y - (\alpha_1 + \alpha_3 - \alpha_4) x};$$

and

$$f_2 := f_2 = \alpha_2 \alpha_3 e^{-\alpha_2 x - (\alpha_1 + \alpha_3 - \alpha_2) y};$$

Provided that $\alpha_1 + \alpha_3 \neq \alpha_2$, the marginal density function of X is, according to Freund,

$$f_X(x) = \left(\frac{\alpha_3}{\alpha_1 + \alpha_3 - \alpha_2} \right) \alpha_2 e^{-\alpha_2 x} + \left(\frac{\alpha_1 - \alpha_2}{\alpha_1 + \alpha_3 - \alpha_2} \right) (\alpha_1 + \alpha_3) e^{-(\alpha_1 + \alpha_3) x}, \quad (2)$$

$$x \geq 0,$$

and provided that $\alpha_1 + \alpha_3 \neq \alpha_4$, the marginal density function of Y is

$$f_Y(y) = \left(\frac{\alpha_1}{\alpha_1 + \alpha_3 - \alpha_4} \right) \alpha_4 e^{-\alpha_4 y} + \left(\frac{\alpha_3 - \alpha_4}{\alpha_1 + \alpha_3 - \alpha_4} \right) (\alpha_1 + \alpha_3) e^{-(\alpha_1 + \alpha_3) y}, \quad (3)$$

$$y \geq 0.$$

We can see that the marginal density functions are, in general, not exponential but rather mixtures of exponential distributions if $\alpha_i > \alpha_{i+1}$ ($i = 1, 3$); otherwise, they are weighted averages. For this reason, this system of distributions should be termed bivariate mixture exponential distributions rather than simply bivariate exponential distributions. The marginal functions $f_X(x)$ and $f_Y(y)$ are exponential distributions only in the special case $\alpha_i = \alpha_{i+1}$ ($i = 1, 3$).

Freund discussed the statistics of the special case when $\alpha_1 + \alpha_3 = \alpha_2 = \alpha_4$ and obtained the following joint density function

$$f(x, y) = \begin{cases} \alpha_1 (\alpha_1 + \alpha_3) e^{-(\alpha_1 + \alpha_3)y} & \text{for } 0 < x < y \\ \alpha_3 (\alpha_1 + \alpha_3) e^{-(\alpha_1 + \alpha_3)x} & \text{for } 0 < y < x \end{cases} \quad (4)$$

with marginal density functions:

$$f_X(x) = (\alpha_1 + \alpha_3 (\alpha_1 + \alpha_3) x) e^{-(\alpha_1 + \alpha_3)x}, \quad x \geq 0 \quad (5)$$

$$f_Y(y) = (\alpha_3 + \alpha_1 (\alpha_1 + \alpha_3) y) e^{-(\alpha_1 + \alpha_3)y}, \quad y \geq 0. \quad (6)$$

The covariance and correlation coefficient of X and Y were derived by Freund, as follows:

$$\text{Cov}(X, Y) = \frac{\alpha_2 \alpha_4 - \alpha_1 \alpha_3}{\alpha_2 \alpha_4 (\alpha_1 + \alpha_3)^2}, \quad (7)$$

$$\rho(X, Y) = \frac{\alpha_2 \alpha_4 - \alpha_1 \alpha_3}{\sqrt{(\alpha_1^2 + 2 \alpha_1 \alpha_3 + \alpha_4^2)(\alpha_3^2 + 2 \alpha_1 \alpha_3 + \alpha_2^2)}}. \quad (8)$$

Note that in general $-\frac{1}{3} < \rho(X, Y) < 1$. The correlation coefficient $\rho(X, Y) \rightarrow 1$ when $\alpha_2, \alpha_4 \rightarrow \infty$, and $\rho(X, Y) \rightarrow -\frac{1}{3}$ when $\alpha_1 = \alpha_3$ and $\alpha_2, \alpha_4 \rightarrow 0$. In many applications, $\alpha_{i+1} > \alpha_i$ ($i = 1, 3$), that is, lifetime tends to be shorter when the other component is out of action. In such cases the correlation is positive.

□ 1.2. Fisher Information Metric

Proposition 1.1 Let F be the set of Freund bivariate mixture exponential distributions, that is,

$$F = \left\{ f : f(x, y; \alpha_i) \right. \\ \left. = \begin{cases} \alpha_1 \alpha_4 e^{-\alpha_4 y - (\alpha_1 + \alpha_3 - \alpha_4)x} & \text{for } 0 < x < y \\ \alpha_3 \alpha_2 e^{-\alpha_2 x - (\alpha_1 + \alpha_3 - \alpha_2)y} & \text{for } 0 < y < x \end{cases}, (i = 1, 2, 3, 4) \right\}. \quad (9)$$

Then we have:

1. Identifying $(\alpha_1, \alpha_2, \alpha_3, \alpha_4)$ as a local coordinate system, F can be regarded as a 4-manifold.
2. F is a Riemannian space and the Fisher information matrix $G = [g_{ij}]$, where

$$g_{ij} = \int_0^\infty \int_0^\infty \frac{\partial^2 \log f}{\partial \alpha_i \partial \alpha_j} \cdot f \, dx \, dy, \quad (i = 1, 2, 3, 4)$$

is given by

$$\begin{aligned}
\text{In}[3]:= & \text{g} = \text{Outer}[\text{FullSimplify}[\int_0^\infty \int_x^\infty -f_1 \text{Simplify}[\frac{\partial^2 \log(f_1)}{\partial \#1 \partial \#2}] dy dx + \\
& \int_0^\infty \int_0^x -f_2 \text{Simplify}[\frac{\partial^2 \log(f_2)}{\partial \#1 \partial \#2}] dx dy, \\
& \{\alpha_1 > 0, \alpha_2 > 0, \alpha_3 > 0, \alpha_4 > 0\}] \&, \\
& \{\alpha_1, \alpha_2, \alpha_3, \alpha_4\}, \{\alpha_1, \alpha_2, \alpha_3, \alpha_4\}] \\
\text{Out}[3]= & \begin{pmatrix} \frac{1}{\alpha_1^2 + \alpha_3 \alpha_1} & 0 & 0 & 0 \\ 0 & \frac{\alpha_3}{\alpha_2^2 (\alpha_1 + \alpha_3)} & 0 & 0 \\ 0 & 0 & \frac{1}{\alpha_3^2 + \alpha_1 \alpha_3} & 0 \\ 0 & 0 & 0 & \frac{\alpha_1}{(\alpha_1 + \alpha_3) \alpha_4^2} \end{pmatrix}
\end{aligned}$$

□ 1.3. α -geometry

In this section we provide the α -connection and various α -curvature objects of the Freund 4-manifold F : the α -curvature tensor, the α -sectional curvatures, the α -mean curvatures, the α -Ricci tensor, and the α -scalar curvature.

1.3.1. α -connection

For each $\alpha \in \mathbb{R}$, the α ($\nabla^{(\alpha)}$)-connection is the torsion-free affine connection with components:

$$\begin{aligned}
\Gamma_{ij,k}^{(\alpha)} = & \int_0^\infty \int_0^\infty \left(\frac{\partial^2 \log(f)}{\partial \alpha_i \partial \alpha_j} \frac{\partial \log(f)}{\partial \alpha_k} + \frac{(1-\alpha)}{2} \frac{\partial \log(f)}{\partial \alpha_i} \frac{\partial \log(f)}{\partial \alpha_j} \frac{\partial \log(f)}{\partial \alpha_k} \right) \cdot f \\
& dx dy, \quad (i, j, k = 1, 2, 3, 4). \quad (10)
\end{aligned}$$

So we have an affine connection ($\nabla^{(\alpha)}$) defined by

$$\langle \nabla_{\partial_i}^{(\alpha)} \partial_j, \partial_k \rangle = \Gamma_{ij,k}^{(\alpha)}, \quad (11)$$

where $g = \langle, \rangle$ is the Fisher information metric.

Note that the 0-connection is the Riemannian connection with respect to the Fisher metric.

Proposition 1.2 The functions $\Gamma_{ij,k}^{(\alpha)}$, ($i, j, k = 1, 2, 3, 4$) are given by

$$\begin{aligned}
\text{In}[4] := & \quad \Gamma^{\alpha} = \text{FullSimplify}[\text{Assuming}[\text{Re}[\alpha_1 + \alpha_3] > 0 \ \&\& \ \text{Re}[\alpha_2] > 0, \\
& \quad \text{Table}[\int_0^{\infty} \int_x^{\infty} f_1 \text{Simplify}[\frac{1}{2} (1 - \alpha) \frac{\partial \log(f_1)}{\partial \alpha_i} \frac{\partial \log(f_1)}{\partial \alpha_j} + \frac{\partial^2 \log(f_1)}{\partial \alpha_i \partial \alpha_j}] \\
& \quad \quad \frac{\partial \log(f_1)}{\partial \alpha_k} \, d y \, d x + \\
& \quad \int_0^{\infty} \int_0^x f_2 \text{Simplify}[\frac{1}{2} (1 - \alpha) \frac{\partial \log(f_2)}{\partial \alpha_i} \frac{\partial \log(f_2)}{\partial \alpha_j} + \frac{\partial^2 \log(f_2)}{\partial \alpha_i \partial \alpha_j}] \\
& \quad \quad \frac{\partial \log(f_2)}{\partial \alpha_k} \, d y \, d x, \{i, 4\}, \{j, 4\}, \{k, 4\}]]];
\end{aligned}$$

For example, the component $\Gamma_{11,1}^{(\alpha)}$ is given by

$$\begin{aligned}
\text{In}[5] := & \quad \Gamma[[1, 1, 1]] \\
\text{Out}[5] = & \quad \frac{2 (\alpha - 1) \alpha_1 - (\alpha + 1) \alpha_3}{2 \alpha_1^2 (\alpha_1 + \alpha_3)^2}
\end{aligned}$$

Proposition 1.3 By solving the equations $\Gamma_{ij,k}^{(\alpha)} = \sum_{b=1}^4 g_{kb} \Gamma_{ij}^b$, ($k = 1, 2, 3, 4$), we obtain the components of $\nabla^{(\alpha)}$:

$$\begin{aligned}
\text{In}[6] := & \quad \text{Table}[\text{FullSimplify}[\text{Solve}[\\
& \quad \Gamma[[i, j, 1]] == \sum_{b=1}^4 g[[1, b]] \Gamma^b(i, j) \wedge \Gamma[[i, j, 2]] == \sum_{b=1}^4 g[[2, b]] \Gamma^b(i, j) \wedge \\
& \quad \Gamma[[i, j, 3]] == \sum_{b=1}^4 g[[3, b]] \Gamma^b(i, j) \wedge \Gamma[[i, j, 4]] == \sum_{b=1}^4 g[[4, b]] \Gamma^b(i, j), \\
& \quad \{\Gamma^1(i, j), \Gamma^2(i, j), \Gamma^3(i, j), \Gamma^4(i, j)\}], \{i, 4\}, \{j, 4\}];
\end{aligned}$$

So we obtain:

$$\begin{aligned}
\text{In}[7] := & \quad \Gamma^1 = \begin{pmatrix} \frac{1}{2} \left(\frac{3\alpha-1}{\alpha_1+\alpha_3} - \frac{\alpha+1}{\alpha_1} \right) & 0 & \frac{\alpha-1}{2(\alpha_1+\alpha_3)} & 0 \\ 0 & \frac{(\alpha+1)\alpha_1\alpha_3}{2\alpha_2^2(\alpha_1+\alpha_3)} & 0 & 0 \\ \frac{\alpha-1}{2(\alpha_1+\alpha_3)} & 0 & \frac{(\alpha+1)\alpha_1}{2\alpha_3(\alpha_1+\alpha_3)} & 0 \\ 0 & 0 & 0 & -\frac{(\alpha+1)\alpha_1\alpha_3}{2(\alpha_1+\alpha_3)\alpha_4^2} \end{pmatrix}; \\
\text{In}[8] := & \quad \Gamma^2 = \begin{pmatrix} 0 & \frac{\alpha-1}{2(\alpha_1+\alpha_3)} & 0 & 0 \\ \frac{\alpha-1}{2(\alpha_1+\alpha_3)} & \frac{\alpha-1}{\alpha_2} & -\frac{(\alpha-1)\alpha_1}{2\alpha_3(\alpha_1+\alpha_3)} & 0 \\ 0 & -\frac{(\alpha-1)\alpha_1}{2\alpha_3(\alpha_1+\alpha_3)} & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix};
\end{aligned}$$

$$\begin{aligned}
\ln[9] := \quad {}^3\Gamma &= \begin{pmatrix} \frac{(\alpha+1)\alpha_3}{2\alpha_1(\alpha_1+\alpha_3)} & 0 & \frac{\alpha-1}{2(\alpha_1+\alpha_3)} & 0 \\ 0 & -\frac{(\alpha+1)\alpha_1\alpha_3}{2\alpha_2^2(\alpha_1+\alpha_3)} & 0 & 0 \\ \frac{\alpha-1}{2(\alpha_1+\alpha_3)} & 0 & \frac{1}{2}\left(\frac{3\alpha-1}{\alpha_1+\alpha_3} - \frac{\alpha+1}{\alpha_3}\right) & 0 \\ 0 & 0 & 0 & \frac{(\alpha+1)\alpha_1\alpha_3}{2(\alpha_1+\alpha_3)\alpha_4^2} \end{pmatrix}; \\
\ln[10] := \quad {}^4\Gamma &= \begin{pmatrix} 0 & 0 & 0 & -\frac{(\alpha-1)\alpha_3}{2\alpha_1(\alpha_1+\alpha_3)} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\alpha-1}{2(\alpha_1+\alpha_3)} \\ -\frac{(\alpha-1)\alpha_3}{2\alpha_1(\alpha_1+\alpha_3)} & 0 & \frac{\alpha-1}{2(\alpha_1+\alpha_3)} & \frac{\alpha-1}{\alpha_4} \end{pmatrix};
\end{aligned}$$

1.3.2 α -curvatures

Proposition 1.4 The components of the α -curvature tensor $R_{ijkl}^{(\alpha)} = g_{bl}(\partial_{\alpha_k} \Gamma_{jk}^b - \partial_{\alpha_j} \Gamma_{ik}^b + \Gamma_{im}^b \Gamma_{jk}^m - \Gamma_{jm}^b \Gamma_{ik}^m)$, $(i, j, k, l = 1, 2, 3, 4)$ are given by

$$\begin{aligned}
\ln[11] := \quad {}^\alpha R &= \text{FullSimplify}\left[\text{Table}\left[\sum_{b=1}^4 g[[b, l]] \text{Simplify}\left[-\frac{\partial \Gamma[[i, k]]}{\partial \alpha_j} + \frac{\partial \Gamma[[j, k]]}{\partial \alpha_i} + \right.\right.\right. \\
&\quad \left.\left.\sum_{m=1}^4 \text{Simplify}\left[\Gamma[[i, m]] \Gamma[[j, k]]^m - \Gamma[[j, m]] \Gamma[[i, k]]^m\right], \right.\right. \\
&\quad \left.\left.\{i, 4\}, \{j, 4\}, \{k, 4\}, \{l, 4\}\right]\right];
\end{aligned}$$

Since the analytical expression for the curvature tensor is very large, we report the components $R_{ijkl}^{(\alpha)}$:

$$\begin{aligned}
\ln[12] := \quad & \text{FullSimplify}\left[\left\{\bar{R}[[1, 2, 1, 2]], \bar{R}[[1, 2, 2, 3]], \bar{R}[[1, 4, 1, 4]], \right.\right. \\
& \left.\left.\bar{R}[[1, 4, 3, 4]], \bar{R}[[2, 3, 2, 3]], \bar{R}[[2, 4, 2, 4]], \bar{R}[[3, 4, 3, 4]]\right\}\right] \\
\text{Out}[12] = \quad & \left\{\frac{(\alpha^2-1)\alpha_3^2}{4\alpha_1\alpha_2^2(\alpha_1+\alpha_3)^3}, \frac{(\alpha^2-1)\alpha_3}{4\alpha_2^2(\alpha_1+\alpha_3)^3}, \frac{(\alpha^2-1)\alpha_3}{4(\alpha_1+\alpha_3)^3\alpha_4^2}, -\frac{(\alpha^2-1)\alpha_1}{4(\alpha_1+\alpha_3)^3\alpha_4^2}, \right. \\
& \left.\frac{(\alpha^2-1)\alpha_1}{4\alpha_2^2(\alpha_1+\alpha_3)^3}, \frac{(\alpha^2-1)\alpha_1\alpha_3}{4\alpha_2^2(\alpha_1+\alpha_3)^2\alpha_4^2}, \frac{(\alpha^2-1)\alpha_1^2}{4\alpha_3(\alpha_1+\alpha_3)^3\alpha_4^2}\right\}
\end{aligned}$$

while the other independent components are zero.

Proposition 1.5 The α -sectional curvatures:

$$\varrho^{(\alpha)}(i, j) = -\frac{R_{ijij}^{(\alpha)}}{(g_{ii}g_{jj} - g_{ij}^2)}, \quad (i, j = 1, 2, 3, 4)$$

are given by

$$\begin{aligned}
In[13] := & \quad \overset{\alpha}{\varrho} = \text{FullSimplify}\left[\text{Table}\left[\text{If}[i == j, 0, -\frac{\overset{\alpha}{R}[i, j, i, j]}{g[i, i] g[j, j] - g[i, j]^2}], \{i, 4\}, \{j, 4\}\right]\right] \\
Out[13] = & \quad \begin{pmatrix} 0 & -\frac{(\alpha^2-1)\alpha_3}{4(\alpha_1+\alpha_3)} & 0 & -\frac{(\alpha^2-1)\alpha_3}{4(\alpha_1+\alpha_3)} \\ -\frac{(\alpha^2-1)\alpha_3}{4(\alpha_1+\alpha_3)} & 0 & -\frac{(\alpha^2-1)\alpha_1}{4(\alpha_1+\alpha_3)} & \frac{1}{4}(1-\alpha^2) \\ 0 & -\frac{(\alpha^2-1)\alpha_1}{4(\alpha_1+\alpha_3)} & 0 & -\frac{(\alpha^2-1)\alpha_1}{4(\alpha_1+\alpha_3)} \\ -\frac{(\alpha^2-1)\alpha_3}{4(\alpha_1+\alpha_3)} & \frac{1}{4}(1-\alpha^2) & -\frac{(\alpha^2-1)\alpha_1}{4(\alpha_1+\alpha_3)} & 0 \end{pmatrix}
\end{aligned}$$

Proposition 1.6 The α -mean curvatures:

$$\varrho^{(\alpha)}(i) = \frac{1}{3} \sum_{j=1, j \neq i}^4 \varrho^{(\alpha)}(i, j), \quad (i = 1, 2, 3, 4)$$

are given by

$$\begin{aligned}
In[14] := & \quad \overset{\alpha}{\varrho} = \text{FullSimplify}\left[\text{Table}\left[\sum_{j=1}^4 \frac{1}{3} \overset{\alpha}{\varrho}[i, j], \{i, 4\}\right]\right] \\
Out[14] = & \quad \left\{-\frac{(\alpha^2-1)\alpha_3}{6(\alpha_1+\alpha_3)}, \frac{1}{6}(1-\alpha^2), -\frac{(\alpha^2-1)\alpha_1}{6(\alpha_1+\alpha_3)}, \frac{1}{6}(1-\alpha^2)\right\}
\end{aligned}$$

Contracting $R_{ijkl}^{(\alpha)}$ with g^{il} , we obtain the components $R_{jk}^{(\alpha)}$ of the Ricci tensor.

Proposition 1.7 The components of the α -Ricci tensor are given by the symmetric matrix $R^{(\alpha)} = [R_{jk}^{(\alpha)}]$.

$$\begin{aligned}
In[15] := & \quad \overset{\alpha}{R} = \text{FullSimplify}\left[\text{Table}\left[\sum_{l=1}^4 \sum_{i=1}^4 g^{-1}[i, l] \overset{\alpha}{R}[i, j, k, l], \{j, 4\}, \{k, 4\}\right]\right] \\
Out[15] = & \quad \begin{pmatrix} -\frac{(\alpha^2-1)\alpha_3}{2\alpha_1(\alpha_1+\alpha_3)^2} & 0 & \frac{\alpha^2-1}{2(\alpha_1+\alpha_3)^2} & 0 \\ 0 & -\frac{(\alpha^2-1)\alpha_3}{2\alpha_2^2(\alpha_1+\alpha_3)} & 0 & 0 \\ \frac{\alpha^2-1}{2(\alpha_1+\alpha_3)^2} & 0 & -\frac{(\alpha^2-1)\alpha_1}{2\alpha_3(\alpha_1+\alpha_3)^2} & 0 \\ 0 & 0 & 0 & -\frac{(\alpha^2-1)\alpha_1}{2(\alpha_1+\alpha_3)\alpha_4^2} \end{pmatrix}
\end{aligned}$$

The α -eigenvalues and the α -eigenvectors of the α -Ricci tensor are given by

$$\begin{aligned}
In[16] := & \quad \text{FullSimplify}\left[\text{Eigenvalues}[\overset{\alpha}{R}]\right] \\
Out[16] = & \quad \left\{0, -\frac{(\alpha^2-1)\alpha_3}{2\alpha_2^2(\alpha_1+\alpha_3)}, \frac{1}{2}(\alpha^2-1)\left(\frac{2}{(\alpha_1+\alpha_3)^2} - \frac{1}{\alpha_1\alpha_3}\right), -\frac{(\alpha^2-1)\alpha_1}{2(\alpha_1+\alpha_3)\alpha_4^2}\right\}
\end{aligned}$$

In[17]:= FullSimplify[Eigenvalues[R]]

$$\text{Out[17]} = \begin{pmatrix} \frac{\alpha_1}{\alpha_3} & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ -\frac{\alpha_3}{\alpha_1} & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Proposition 1.8 The manifold F has a constant α -scalar curvature $R^{(\alpha)} = g^{ij} R_{ij}^{(\alpha)}$.

In[18]:= $R = \text{FullSimplify}[\sum_{j=1}^4 \sum_{i=1}^4 g^{-1}[i, j] R[i, j]]$

$$\text{Out[18]} = -\frac{3}{2} (\alpha^2 - 1)$$

Note that the Freund manifold F has a positive constant scalar curvature $R^{(0)} = \frac{3}{2}$ when $\alpha = 0$, so geometrically it constitutes part of a sphere (compare, for example, Kobayashi and Nomizu [9]).

■ 2. Submanifolds of the Freund 4-Manifold F

We consider four submanifolds F_i ($i = 1, 2, 3, 4$) of the 4-manifold F of Freund bivariate exponential distributions (1) $f(x, y; \alpha_1, \alpha_2, \alpha_3, \alpha_4)$, which includes the case of statistically independent random variables. It also includes the special case of an Absolutely Continuous Bivariate Exponential Distribution called ACBED (or ACBVE) by Block and Basu (compare Hutchinson and Lai [10]). We use the coordinate system $(\alpha_1, \alpha_2, \alpha_3, \alpha_4)$ for the submanifolds F_i , ($i \neq 4$) and the coordinate system $(\lambda_1, \lambda_{12}, \lambda_2)$ for ACBED of the Block and Basu case.

□ 2.1. Submanifold $F_1 \subset F: \alpha_2 = \alpha_1$ and $\alpha_4 = \alpha_3$

The distributions are of the form:

$$f(x, y; \alpha_1, \alpha_3) = f_1(x; \alpha_1) f_2(y; \alpha_3), \quad (12)$$

where f_i are the density functions of the one-dimensional exponential distributions with the parameters $\alpha_i > 0$ ($i = 1, 3$). This is the case for statistical independence of X and Y , so the space F_1 is the direct product of two Riemannian spaces:

$$\{f_1(x; \alpha_1) : f_1(x; \alpha_1) = \alpha_1 e^{-\alpha_1 x}, \alpha_1 > 0\}$$

and

$$\{f_2(y; \alpha_3) : f_2(y; \alpha_3) = \alpha_3 e^{-\alpha_3 y}, \alpha_3 > 0\}.$$

Proposition 2.1 The Fisher information metric $[g_{ij}]$ is given by

$$[g_{ij}] = \begin{pmatrix} \frac{1}{\alpha_1^2} & 0 \\ 0 & \frac{1}{\alpha_3^2} \end{pmatrix}. \quad (13)$$

Proposition 2.2 The components of $\nabla^{(\alpha)}$ are given by

$$\Gamma^1_{11,1} = \frac{\alpha-1}{\alpha_1^3}, \quad \Gamma^1_{22,2} = \frac{\alpha-1}{\alpha_3^3}, \quad \Gamma^1_{11} = \frac{\alpha-1}{\alpha_1}, \quad \Gamma^2_{22} = \frac{\alpha-1}{\alpha_3}, \quad (14)$$

while the other components are zero.

Proposition 2.3 The α -curvature tensor, α -Ricci tensor, and α -scalar curvature of F_1 are zero.

□ 2.2. Submanifold $F_2 \subset F: \alpha_3 = \alpha_1$ and $\alpha_4 = \alpha_2$

The distributions are of the form:

$$f(x, y) = \begin{cases} \alpha_1 \alpha_2 e^{-\alpha_2 y - (2\alpha_1 - \alpha_2)x} & \text{for } 0 < x < y \\ \alpha_1 \alpha_2 e^{-\alpha_2 x - (2\alpha_1 - \alpha_2)y} & \text{for } 0 < y < x \end{cases} \quad (15)$$

with parameters $\alpha_1, \alpha_2 > 0$. The covariance, correlation coefficient, and marginal functions of X and Y are given by

$$\text{Cov}(X, Y) = \frac{1}{4} \left(\frac{1}{\alpha_1^2} - \frac{1}{\alpha_2^2} \right), \quad (16)$$

$$\rho(X, Y) = 1 - \frac{4\alpha_1^2}{3\alpha_1^2 + \alpha_2^2}, \quad (17)$$

$$f_X(x) = \left(\frac{\alpha_1}{2\alpha_1 - \alpha_2} \right) \alpha_2 e^{-\alpha_2 x} + \left(\frac{\alpha_1 - \alpha_2}{2\alpha_1 - \alpha_2} \right) (2\alpha_1) e^{-2\alpha_1 x}, \quad x \geq 0 \quad (18)$$

$$f_X(x) = \left(\frac{\alpha_1}{2\alpha_1 - \alpha_2} \right) \alpha_2 e^{-\alpha_2 x} + \left(\frac{\alpha_1 - \alpha_2}{2\alpha_1 - \alpha_2} \right) (2\alpha_1) e^{-2\alpha_1 x}, \quad x \geq 0. \quad (19)$$

Note that $\rho(X, Y) = 0$ when $\alpha_1 = \alpha_2$.

F_2 forms an exponential family, with natural parameters (α_1, α_2) and the potential function

$$\ln[\eta] := \psi = -\log(\alpha_1) - \log(\alpha_2);$$

Proposition 2.4 The submanifold F_2 is an isometric diffeomorph of the submanifold F_1 .

Proof: Since ψ is a potential function, the Fisher information metric is given by the Hessian of ψ , that is,

$$g_{ij} = \frac{\partial^2 \psi}{\partial \alpha_i \partial \alpha_j}, \quad (i = 1, 2). \quad (20)$$

Then we have the Fisher information metric (10) by a straightforward calculation.

2.2.1 Mutually Dual Foliations

Since $\nabla_{\partial_i}^{(1)} \partial_j = 0$, (α_1, α_2) is a 1-affine coordinate system, and the (-1) -affine coordinate system is given by

$$\text{In}[20]:= \eta = \text{Outer}[\text{FullSimplify}[\frac{\partial \psi}{\partial \#1}, \{\alpha_1 > 0, \alpha_2 > 0\}] \&, \{\alpha_1, \alpha_2\}]$$

$$\text{Out}[20]= \left\{ -\frac{1}{\alpha_1}, -\frac{1}{\alpha_2} \right\}$$

These coordinates have the potential function

$$\text{In}[21]:= \phi = \text{FullSimplify}[\sum_{i=1}^2 \eta[i] \alpha_i - \psi]$$

$$\text{Out}[21]= \log(\alpha_1) + \log(\alpha_2) - 2$$

So the coordinates (α_1, α_2) and $(-\frac{1}{\alpha_1}, -\frac{1}{\alpha_2})$ are mutually dual with respect to the Fisher metric, and the tetrad $\{F_2, g, \nabla^{(1)}, \nabla^{(-1)}\}$ is a dually flat space. Therefore, F_2 has dually orthogonal foliations.

For example, take (α_1, η_2) as a coordinate system for F_2 ; then

$$f(x, y) = \begin{cases} -\frac{\alpha_1}{\eta_2} e^{(\frac{1}{\eta_2})y - (2\alpha_1 + \frac{1}{\eta_2})x} & \text{for } 0 < x < y \\ -\frac{\alpha_1}{\eta_2} e^{(\frac{1}{\eta_2})x - (2\alpha_1 + \frac{1}{\eta_2})y} & \text{for } 0 < y < x \end{cases} \quad (21)$$

and the Fisher information metric is

$$\begin{pmatrix} -\frac{1}{\alpha_1^2} & 0 \\ 0 & \frac{1}{(\eta_2)^2} \end{pmatrix}. \quad (22)$$

2.2.2 Neighbourhoods of Independence in F_2

An important practical application of the Freund submanifold F_2 is the representation of a bivariate stochastic process for which the marginals are identical exponentials. The next result is important because it provides a topological neighbourhood of that subspace W in F_2 consisting of the bivariate processes that have zero covariance: we obtain a neighbourhood of independence for random, that is, exponentially distributed processes.

Proposition 2.5 Let $\{F_2, g, \nabla^{(1)}, \nabla^{(-1)}\}$ be the manifold F_2 with Fisher metric g and exponential connection $\nabla^{(1)}$. Then F_2 can be realized in $\mathbb{R}^3$ by the graph of a potential function, namely, F_2 can be realized by the affine immersion $\{b, \xi\}$

$$b : F_2 \rightarrow \mathbb{R}^3 : \begin{pmatrix} \alpha_1 \\ \alpha_2 \end{pmatrix} \mapsto \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \psi \end{pmatrix}, \quad \xi = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \quad (23)$$

where $\psi = -\log(\alpha_1) - \log(\alpha_2)$ and ξ is the transversal vector field along b .

In F_2 , the submanifold W consisting of the independent case ($\alpha_1 = \alpha_2$) is represented by the curve

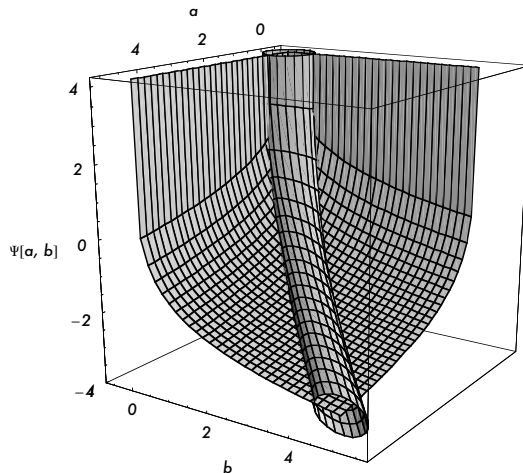
$$(0, \infty) \rightarrow \mathbb{R}^3 : (\alpha_1, \alpha_1) \mapsto (\alpha_1, \alpha_1, -2 \log(\alpha_1)), \quad \xi = (0, 0, 1). \quad (24)$$

This is illustrated in the graphic that shows S , an affine embedding of F_2 as a surface in $\mathbb{R}^3$, and T , an $\mathbb{R}^3$ -tubular neighbourhood of W , as the curve $\alpha_1 = \alpha_2$ in the surface. This curve represents all bivariate distributions having identical exponential marginals and zero covariance; its tubular neighbourhood represents departures from independence. We parametrize here with $a = \alpha_1, b = \alpha_2$.

```
In[22]:= S = ParametricPlot3D[Evaluate[{a, b, -Log(a) - Log(b)}], {a, 0.00001, 5},
    {b, 0.00001, 5}, ViewPoint -> {3, 2, 1}, BoxRatios -> {1, 1, 1},
    AxesLabel -> {a, b, Psi(a, b)}, DisplayFunction -> Identity];

T = ParametricPlot3D[{a -  $\frac{0.6 \text{Cos}[\text{theta}]}{\sqrt{1 + a^2}}$ , a - 0.6 Sin[theta],
    -  $\frac{0.6 a \text{Cos}[\text{theta}]}{\sqrt{1 + a^2}}$  - 2 Log[a]} // Evaluate,
    {a, 0.00001, 5}, {theta, 0, 2 Pi}, PlotPoints -> {20, 20},
    ViewPoint -> {3, 2, 1}, BoxRatios -> {1, 1, 1},
    AxesLabel -> {a, b, Psi(a, b)}, DisplayFunction -> Identity];

In[24]:= Show[{S, T}, DisplayFunction -> $DisplayFunction]
```



□ **2.3. Submanifold $F_3 \subset F$: $\alpha_2 = \alpha_4 = \alpha_1 + \alpha_3$**

The distributions are of the form:

$$f(x, y; \alpha_1, \alpha_2) = \begin{cases} \alpha_1 (\alpha_1 + \alpha_3) e^{-(\alpha_1 + \alpha_3)y} & \text{for } 0 < x < y \\ \alpha_3 (\alpha_1 + \alpha_3) e^{-(\alpha_1 + \alpha_3)x} & \text{for } 0 < y < x \end{cases} \quad (25)$$

with parameters $\alpha_1, \alpha_3 > 0$. The covariance, correlation coefficient, and marginal functions of X and Y are given by

$$\text{Cov}(X, Y) = \frac{\alpha_1^2 + \alpha_1 \alpha_3 + \alpha_3^2}{(\alpha_1 + \alpha_3)^4}, \quad (26)$$

$$\rho(X, Y) = \frac{\alpha_1^2 + \alpha_1 \alpha_3 + \alpha_3^2}{\sqrt{2(\alpha_1 + \alpha_3)^2 - \alpha_1^2} \sqrt{2\alpha_1^2 + 4\alpha_1 \alpha_3 + \alpha_3^2}}, \quad (27)$$

$$f_X(x) = (\alpha_3 (\alpha_1 + \alpha_3)x + \alpha_1) e^{-(\alpha_1 + \alpha_3)x}, \quad x \geq 0 \quad (28)$$

$$f_Y(y) = (\alpha_1 (\alpha_1 + \alpha_3)y + \alpha_3) e^{-(\alpha_1 + \alpha_3)y}, \quad y \geq 0. \quad (29)$$

Note that the correlation coefficient is positive.

Proposition 2.6 The Fisher information metric $[g_{ij}]$ is given by

$$[g_{ij}] = \begin{pmatrix} \frac{2\alpha_1 + \alpha_3}{\alpha_1(\alpha_1 + \alpha_3)^2} & \frac{1}{(\alpha_1 + \alpha_3)^2} \\ \frac{1}{(\alpha_1 + \alpha_3)^2} & \frac{\alpha_1 + 2\alpha_3}{\alpha_3(\alpha_1 + \alpha_3)^2} \end{pmatrix}. \quad (30)$$

Proposition 2.7 The α -connection components of F_3 are given by

$$\begin{aligned} \Gamma_{11,1}^{(\alpha)} &= \frac{4(\alpha - 1)\alpha_1^2 + (\alpha - 3)\alpha_3\alpha_1 - (\alpha + 1)\alpha_3^2}{2\alpha_1^2(\alpha_1 + \alpha_3)^3}, \\ \Gamma_{11}^1 &= \frac{(\alpha_1 + 2\alpha_3)(4(\alpha - 1)\alpha_1^2 + (\alpha - 3)\alpha_3\alpha_1 - (\alpha + 1)\alpha_3^2)}{4\alpha_1(\alpha_1 + \alpha_3)^3}, \\ \Gamma_{11}^2 &= \frac{\alpha_3(-4(\alpha - 1)\alpha_1^2 - (\alpha - 3)\alpha_3\alpha_1 + (\alpha + 1)\alpha_3^2)}{4\alpha_1(\alpha_1 + \alpha_3)^3}, \end{aligned} \quad (31)$$

while the other independent components are zero.

Proposition 2.8 The α -curvature tensor, α -Ricci tensor, and α -scalar curvature of F_3 are zero.

□ 2.4. Submanifold $F_4 \subset F$: ACBED of Block and Basu

The distributions are of the form:

$$f(x, y; \lambda_1, \lambda_{12}, \lambda_2) = \begin{cases} \frac{\lambda_1 \lambda (\lambda_2 + \lambda_{12})}{\lambda_1 + \lambda_2} e^{-\lambda_1 x - (\lambda_2 + \lambda_{12}) y} & \text{for } 0 < x < y \\ \frac{\lambda_2 \lambda (\lambda_1 + \lambda_{12})}{\lambda_1 + \lambda_2} e^{-(\lambda_1 + \lambda_{12}) x - \lambda_2 y} & \text{for } 0 < y < x, \end{cases} \quad (32)$$

where the parameters $\lambda_1, \lambda_{12}, \lambda_2$ are positive, and $\lambda = \lambda_1 + \lambda_2 + \lambda_{12}$. This distribution was derived originally by omitting the singular part of Marshall and Olkin's distribution (compare [10]); Block and Basu called it the ACBED to emphasize that it is an absolutely continuous part of a bivariate exponential distribution. Alternatively, it can be derived from the Freund distribution (1), with

$$\begin{aligned} \alpha_1 &= \lambda_1 + \frac{\lambda_1 \lambda_{12}}{\lambda_1 + \lambda_2}, \\ \alpha_2 &= \lambda_1 + \lambda_{12}, \\ \alpha_3 &= \lambda_2 + \frac{\lambda_2 \lambda_{12}}{\lambda_1 + \lambda_2}, \\ \alpha_4 &= \lambda_2 + \lambda_{12}. \end{aligned}$$

By substitution we obtained the covariance, correlation coefficient, and marginal functions of X and Y :

$$\text{Cov}(X, Y) = \frac{(\lambda_1 + \lambda_2)^2 (\lambda_1 + \lambda_{12}) (\lambda_2 + \lambda_{12}) - \lambda^2 \lambda_1 \lambda_2}{\lambda^2 (\lambda_1 + \lambda_2)^2 (\lambda_1 + \lambda_{12}) (\lambda_2 + \lambda_{12})}, \quad (33)$$

$$\begin{aligned} \rho(X, Y) &= ((\lambda_1 + \lambda_2)^2 (\lambda_1 + \lambda_{12}) (\lambda_2 + \lambda_{12}) - \lambda^2 \lambda_1 \lambda_2) / \\ &\quad \sqrt{(((\lambda_1 + \lambda_2)^2 (\lambda_2 + \lambda_{12})^2 + \lambda_1 \lambda^2 (\lambda_1 + 2 \lambda_2)) \\ &\quad ((\lambda_1 + \lambda_2)^2 (\lambda_1 + \lambda_{12})^2 + \lambda_2 \lambda^2 (\lambda_2 + 2 \lambda_1)))}, \end{aligned} \quad (34)$$

$$f_X(x) = \left(\frac{-\lambda_{12}}{\lambda_1 + \lambda_2} \right) \lambda e^{-\lambda x} + \left(\frac{\lambda}{\lambda_1 + \lambda_2} \right) (\lambda_1 + \lambda_{12}) e^{-(\lambda_1 + \lambda_{12}) x}, \quad x \geq 0 \quad (35)$$

$$f_Y(y) = \left(\frac{-\lambda_{12}}{\lambda_1 + \lambda_2} \right) \lambda e^{-\lambda y} + \left(\frac{\lambda}{\lambda_1 + \lambda_2} \right) (\lambda_2 + \lambda_{12}) e^{-(\lambda_2 + \lambda_{12}) y}, \quad y \geq 0. \quad (36)$$

Note that the correlation coefficient is positive, and the marginal functions are a negative mixture of two exponentials.

The Christoffel symbols, curvature tensor, Ricci tensor, sectional curvatures, mean curvatures, and scalar curvature were computed, but are not listed because they are somewhat cumbersome.

When $\lambda_2 = \lambda_1$, this family of distributions becomes an exponential family with natural parameters $(\lambda_1, \lambda_{12})$ and potential function $\psi = \log(2) - \log(\lambda_1 + \lambda_{12}) - \log(2\lambda_1 + \lambda_{12})$, so it would be easy to derive the α -geometry, for example,

$$[g_{ij}] = \begin{pmatrix} \frac{4}{(2\lambda_1 + \lambda_{12})^2} + \frac{1}{(\lambda_1 + \lambda_{12})^2} & \frac{2}{(2\lambda_1 + \lambda_{12})^2} + \frac{1}{(\lambda_1 + \lambda_{12})^2} \\ \frac{2}{(2\lambda_1 + \lambda_{12})^2} + \frac{1}{(\lambda_1 + \lambda_{12})^2} & \frac{1}{(2\lambda_1 + \lambda_{12})^2} + \frac{1}{(\lambda_1 + \lambda_{12})^2} \end{pmatrix}, \quad (37)$$

and the $\nabla^{(\alpha)}$ components are as follows:

$$\frac{1}{\Gamma} = \begin{pmatrix} \frac{1-\alpha}{\lambda_1 + \lambda_{12}} + \frac{4(\alpha-1)}{2\lambda_1 + \lambda_{12}} & \frac{(\alpha-1)\lambda_{12}}{(\lambda_1 + \lambda_{12})(2\lambda_1 + \lambda_{12})} \\ \frac{(\alpha-1)\lambda_{12}}{(\lambda_1 + \lambda_{12})(2\lambda_1 + \lambda_{12})} & -\frac{(\alpha-1)\lambda_1}{(\lambda_1 + \lambda_{12})(2\lambda_1 + \lambda_{12})} \end{pmatrix}, \quad (38)$$

$$\frac{2}{\Gamma} = \begin{pmatrix} -\frac{2(\alpha-1)\lambda_{12}}{(\lambda_1 + \lambda_{12})(2\lambda_1 + \lambda_{12})} & \frac{2(\alpha-1)\lambda_1}{(\lambda_1 + \lambda_{12})(2\lambda_1 + \lambda_{12})} \\ \frac{2(\alpha-1)\lambda_1}{(\lambda_1 + \lambda_{12})(2\lambda_1 + \lambda_{12})} & \frac{1-\alpha}{2\lambda_1 + \lambda_{12}} + \frac{2(\alpha-1)}{\lambda_1 + \lambda_{12}} \end{pmatrix}, \quad (39)$$

while the α -curvatures vanish.

■ 3. Concluding Remarks

We have derived the information geometry of the 4-manifold F of Freund bivariate mixture exponential distributions, which admits positive and negative covariance. The α -curvature objects are derived for F and those on four submanifolds, including the case of statistically independent random variables and the special case ACBED of Block and Basu. We provide examples of neighbourhoods of the independent case for bivariate distributions having identical exponential marginals and zero covariance. The Freund manifold has a constant 0-scalar curvature, so geometrically it constitutes part of a sphere.

■ Acknowledgment

The authors wish to thank the referees for suggesting improvements and the Libyan Ministry of Education for a scholarship for Arwini.

■ References

- [1] K. Arwini and C. T. J. Dodson, "Information-Geometric Neighbourhoods of Randomness and Geometry of the McKay Bivariate Gamma 3-Manifold." *Indian Journal of Statistics*, **66**(2), 2004 pp. 211–231. www.ma.umist.ac.uk/kd/PREPRINTS/gamran.pdf.

- [2] S.-I. Amari, O. E. Barndorff-Neilsen, R. E. Kass, S. L. Lauritzen, and C. R. Rao, *Differential-Geometrical Methods in Statistics*, Springer Lecture Notes in Statistics **28**, Berlin: Springer-Verlag, 1985.
- [3] S.-I. Amari and H. Nagaoka, *Methods of Information Geometry*, Transactions of Mathematical Monographs, Vol. 191, American Mathematical Society, 2000.
- [4] C. T. J. Dodson and H. Matsuzoe, "An Affine Embedding of the Gamma Manifold," *Applied Sciences* [online], **5**(1), 2003 pp. 1–6.
www.ma.umist.ac.uk/kd/PREPRINTS/affimm.pdf.
- [5] Y. Cai, C. T. J. Dodson, A. J. Doig, and O. Wolkenhauer, "Information-Theoretic Analysis of Protein Sequences Shows That Amino Acids Self-Cluster," *Journal of Theoretical Biology*, **218**(4), 2002 pp. 409–418.
- [6] C. T. J. Dodson, "Spatial Statistics and Information Geometry for Parametric Statistical Models of Galaxy Clustering," *International Journal of Theoretical Physics*, **38**(10), 1999 pp. 2585–2597.
- [7] C. T. J. Dodson, "Geometry for Stochastically Inhomogeneous Spacetimes," *Nonlinear Analysis*, **47**(5), 2001 pp. 2951–2958.
- [8] J. E. Freund, "A Bivariate Extension of the Exponential Distribution," *Journal of the American Statistical Association*, **56**, 1961 pp. 971–977.
- [9] S. Kobayashi and K. Nomizu, *Foundations Of Differential Geometry*, Vol. 1, New York: John Wiley & Sons, 1963 p. 294.
- [10] T. P. Hutchinson and C. D. Lai, *Continuous Multivariate Distributions, Emphasizing Applications*, Adelaide: Rumsby Scientific Publishing, 1990.

Khadiga Arwini

C. T. J. Dodson

School of Mathematics

Manchester University

Sackville Street

Manchester M601QD, UK

arwini2001@yahoo.com

ctdodson@manchester.ac.uk

www.ma.umist.ac.uk/kd/homepage/dodson.html

Article Submissions to The Mathematica Journal

The Mathematica Journal publishes articles and notes on all aspects of *Mathematica* and related subjects. Its goal is to inform and challenge the community of *Mathematica* users and to enable readers at all levels of proficiency to use *Mathematica* more effectively. A high standard of *Mathematica* programming, using clarity and conciseness in writing, is valued. *The Mathematica Journal* reserves the right to edit contributions, with changes subject to the authors' approval.

The Mathematica Journal also publishes *Mathematica* notebooks and packages, as well as programs in other languages that can be used to complement *Mathematica*. All programs are accompanied by a note or article. Programs distributed through the journal can be used freely and duplicated for noncommercial purposes.

Some of the categories of material published in *The Mathematica Journal* are listed below.

- Reports of applications of *Mathematica* to particular fields in education, research, or business. The level of exposition should be such that readers from other fields can grasp the essence of the application, the reasons why *Mathematica* is being used, and the possible relevance to other fields of the techniques being employed.
- Descriptions of programs—*Mathematica* packages and notebooks, or *Mathematica*-related programs in other languages—whether or not accompanied by the code itself. The programs should be exemplary in their implementation and thoroughly tested.
- *Mathematica* programming advice. This can range from notes about little-known programming tricks to tutorials focusing on a general problem or technique. The main test of applicability here is how many readers are likely to benefit from the ideas, but even a relatively esoteric application can aid in the teaching of good programming principles.
- Discussion of issues beyond *Mathematica*. Essays, columns, and surveys on subjects that do not involve *Mathematica* directly may be considered, if they are likely to be of general interest to the *Mathematica* community. Possible subjects include symbolic computation, numerical computation, computer-based education, computer graphics, computer art, and so forth. Articles of this type should be particularly well written and well

argued, and their relevance to the *Mathematica* community should be clear.

- General informational articles, profiles of exemplary *Mathematica* users or applications, interviews, and news related to the *Mathematica* community will also be sought for publication online on a frequent basis.

To submit a contribution, send the material—together with a cover letter stating that it is being submitted for publication in *The Mathematica Journal*—in one of the following formats. Use filenames that suggest their contents.

Web

Articles, graphics, and other files may be uploaded on the Internet at www.mathematica-journal.com/upload.

Email

Articles and other material may be sent by email to submissions@mathematica-journal.com. Do not email articles or attachments that exceed 1MB.

Hard Copy and/or Physical Media by Mail

Send to:

Editor
The Mathematica Journal
PO Box 6359
Champaign, IL 61826-6359 USA

Articles, packages, graphics, notebooks, and programs submitted for publication can be sent on cdrom, zip, or other disk. Submissions will not be returned. Do not send proprietary or confidential information.

■ Manuscript Submission Guidelines

Manuscripts must be in English. Manuscripts should be submitted as properly formatted notebooks made with *Mathematica* version 3 or later. Notebooks should be well structured, making good use of sections, text, and inline and displayed equations (if required), and should use the *Mathematica* style sheet “TMJv4Style.nb”. This style sheet along with an authoring palette can be downloaded from www.mathematica-journal.com/submissions.

Manuscripts should present a significant application of *Mathematica*. Manuscripts under consideration by or submitted simultaneously to other journals will not be considered.

Remember to include the following:

1. the title, if the title has more than 50 characters, submit an alternative short title;
2. your name(s), affiliation, postal, and electronic mail address;
3. an abstract of at most 150 words.

The remaining material should be included in the following order:

1. text of the article or note;
2. references (see below);
3. a list of additional electronic material, including, if appropriate, information on the necessary environment (machine, operating system, and so on);
4. a biography of each author, limited to 60 words per author, and 180 words overall.

References should be numbered in the text in order of appearance. In the listing at the end of the article, each reference should include full information: author or institution, full title, publisher, city, and year (for books, manuals, etc.); or full journal name, volume, year, and page range (for papers). References to software and hardware should contain complete manufacturer's or distributor's names and addresses. See format in a current issue.

All references in the bibliography should be cited in the text, or accompanied by comments stating their relevance.

Avoid footnotes if possible. If used, they should be numbered sequentially and clearly marked in the text, with the footnotes as a section at the end of the article.

Please observe the following guidelines for text and embedded code.

- Use of the word *Mathematica* should be kept to a minimum, since the reference is almost always clear from the context. It is especially discouraged in titles.
- Technical terms unlikely to be known to nonspecialists should be italicized the first time they are used, and, unless the first use is self-explanatory, they should be defined or explained then. Italics should not be used for emphasis, except very occasionally.
- *Mathematica* words, symbols, or expressions should appear in a font distinct from that used for text. The use of a fixed-pitch font, in which all characters have the same width, is strongly encouraged. Use the author's palette and *The Mathematica Journal* style sheet.
- References to *Mathematica* functions should not be followed by [], unless what is meant is that the function is being invoked with no arguments.
- Lines of displayed code, whether from packages or interactive sessions, should be no longer than 60 characters.
- Do not use *Mathematica* comments in interactive sessions; instead, alternate explanatory text using complete sentences with input/output pairs. It is especially important to explain the commands in a long interactive session as you go along, rather than all at once at the beginning or at the end of the session. As a rule, notebooks should be cast into this same format: explanatory text alternating with code.
- Always mention which version of *Mathematica* you are running.

■ Guidelines for *Mathematica* Packages

Packages are programs written in *Mathematica*, to be used on their own or integrated with the user's code.

The single most important consideration in designing a *Mathematica* package for distribution is that it must be readily usable by others. Clarity is the goal and a good test of the clarity of your design is the ease with which it can be documented. Ideally, the purpose and use of each symbol visible to the user should be simple enough to explain in its usage string.

The more simply your design can be described, the more likely people are to be able to understand and use what you have built. In addition, such designs are often more general and therefore usable in a wider range of circumstances.

The book *Programming in Mathematica*, Third Edition, by Roman Maeder (Addison-Wesley, 1996) provides good pointers on programming style.

Classifieds

Classified advertising is provided for *Mathematica* users. Rates are available at www.mathematica-journal.com/advertising.

Mathematica Training and Consulting—Whether you are an individual or a company seeking *Mathematica* training, Wolfram Education Group provides a learning solution with **Certified Training**. Register for classes and review the Education Benefits Program for corporate account training incentives and discounts. If you are looking for **Independent Training** for additional courses, workshops, or tutorials, the Wolfram Education Group offers a variety of learning opportunities from independent trainers and companies with years of *Mathematica* experience. Wolfram Research also offers direct **Consulting Services**, such as reviewing and debugging your *Mathematica* programs, isolating problems in large *Mathematica* programs, implementing custom features, and writing complete programs based on user specifications.

More info: www.wolfram.com/services/training

Online Calculus Solver—calc101.com supplies automatic step-by-step solutions for derivatives, elementary integrals, matrix algebra, and polynomial operations.

More info: www.calc101.com



Are you up to the *Mathematica*[®] challenge?

Wolfram Research's *Mathematica* technology stands behind many of the most important scientific discoveries and technical innovations of the last two decades. In our continuing commitment to revolutionizing the technology and science of computation, we now have unique new opportunities in our world-class team at both junior and senior levels.

If you're skilled with *Mathematica*, clear-thinking, pragmatic, and excited about transforming the way today's most innovative people work, go to **career.wolfram.com** and get ready for the challenge of a lifetime.

An Equal Opportunity Employer



WOLFRAMRESEARCH

New Tools • New Science • New Technology

