

The *Mathematica*[®] Journal

Volume 9, Issue 2

Contents

Departments

Letters	254
Staff and Contributors	255
Calendar	256
News Bulletins	264
New Products	267
New Resources	272
New Publications	274
Article Submissions to <i>The Mathematica Journal</i>	496
Classifieds	500

Columns

Tricks of the Trade <i>edited by Paul Abbott</i>	281
In and Out <i>edited by Paul Abbott</i>	298
Trott's Corner <i>by Michael Trott</i>	314

Feature Articles

Learning about Differential Equations from their Symmetries <i>by Scott A. Herod</i>	334
p -adic Arithmetic <i>by Stany De Smedt</i>	349
Updating a Geographic Database <i>by Leendert van Gastel and Harry Uitermark</i>	358
A Generator of Rook Polynomials <i>by Daniel C. Fielder</i>	371
Customizing the Help Browser <i>by Pavi Sandhu</i>	376
The Second Set of Magic Angles of Projectile Motion <i>by Haiduke Sarafian</i>	388
Spatial Inversion: Reflective Anamorphograms <i>by Philip W. Kuchel</i>	392
Path Integral Methods for Parabolic Partial Differential Equations with Examples from Computational Finance <i>by Andrew Lyasoff</i>	399
AuthorTools: A Package for Document Processing <i>by Pavi Sandhu</i>	423
Integral Equations <i>by Stan Richardson</i>	460
Algebraic Programming in <i>Mathematica</i> <i>by Andrzej Kozłowski</i>	483

The electronic version of this issue is available at www.mathematica-journal.com/issue/v9i2.

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

On the Cover

Sándor Kabai used *Mathematica* to generate the cover design. His book *Mathematical Graphics I: Lessons in Computer Graphics Using Mathematica* was published in Hungary by Uniconstant in 2002. To view more of Kabai's work, see: www.kabai.hu.

Letters

Text in 3D Graphics

I read with great interest your answer about a way to include text in 3D graphics in *The Mathematica Journal*, 9(1), 2003

(www.mathematica-journal.com/issue/v9i1/contents/Inout9-1/Inout9-1_4.html).

You correctly pointed out that the package by Mr. Roland Jakschewitz uses the public domain Hershey fonts to produce text primitives. That package, however, is limited to producing text in 2D graphics. Your readers might be interested to know that a package that produces graphics primitives for text in 3D graphics does exist. In the package Text3D.m, I have adapted the package of Mr. Jakschewitz for the creation and placement of text based on the Hershey fonts as 3D primitives. The package and a demonstration notebook are available in *MathSource*® at library.wolfram.com/infocenter/MathSource/4535. You can also see some implementations in my gallery of animated 3D graphics at www.iulaw.indy.indiana.edu/instructors/Georgakopoulos/Prof/all3dgraphs.htm.

Nicholas L. Georgakopoulos
Carmel, Indiana

Staff and Contributors

Managing Editor: George Beck

Project Manager: Louise Holubek

Design Team: Jeremy Davis, Megan Gillette

Compositor: Glenn Scholebo

Archives and Web Production: Jean Buck

Web Development: Andy Hunt, Lori Goodman

Editorial Staff: George Beck, Marcia Krause, Larry Adelston

Customer Service: Mandy Pope

Columnists: Paul Abbott, Michael Trott

Contributors: Stany De Smedt, Daniel C. Fielder, Scott A. Herod, Andrzej Kozłowski, Philip W. Kuchel, Andrew Lyasoff, Stan Richardson, Pavi Sandhu, Haiduke Sarafian, Harry Uitermark, Leendert van Gastel

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); Alfred van der Poorten (Macquarie University); Bill Press (Harvard University); Sam Savage (University of Chicago); Dana Scott (Carnegie Mellon University); Lawrence M. Seiford (University of Massachusetts, Amherst); Lynn Steen (St. Olaf College); Francis Sullivan (Institute for Defense Analyses); Bill Thurston (University of California, Davis); Michael Trott (Wolfram Research); 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.

© 2004 Wolfram Media, Inc. All rights reserved. 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 used herein 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

The calendar lists professional meetings, conferences, trade shows, exhibits, and *Mathematica* training courses.

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

May

May, 2004

M220: Programming in *Mathematica*

Wolfram Education Group

May 6–7, Chicago, Illinois

Info: www.wolfram.com/services/education/calendar.cgi

M101: A First Course in *Mathematica*

Wolfram Education Group

May 17–18, New York, New York

May 17–18, San Francisco, California

Info: www.wolfram.com/services/education/calendar.cgi

May 3–4, 2004

M101: A First Course in *Mathematica*

Mathematica Nordic Education Centre

Aarhus, Denmark

Info: www.wolfram.com/services/education/calendar.cgi

May 11, 2004

Computeralgebrasimulation mit *Mathematica*

ADDITIVE GmbH

Frankfurt, Germany

Info: www.additive-net.de/software/workshops/agenda_frankfurt.phtml

May 12, 2004

Computeralgebrasimulation mit *Mathematica*

ADDITIVE GmbH

Köln, Germany

Info: www.additive-net.de/software/workshops/agenda_koeln.phtml

May 13, 2004

Computeralgebrasimulation mit *Mathematica*

ADDITIVE GmbH

Stuttgart, Germany

Info: www.additive-net.de/software/workshops/agenda_stuttgart.phtml

May 13–14, 2004

M110: Working with Data Using *Mathematica* Online Class

Wolfram Education Group

Info: www.wolfram.com/services/education/calendar.cgi

May 24–25, 2004

M220: Programming in *Mathematica*

Mathematica Nordic Education Centre

Copenhagen, Denmark

Info: www.wolfram.com/services/education/calendar.cgi

June

June, 2004

M101: A First Course in *Mathematica*

Wolfram Education Group

June 3–4, Grenoble, France

June 3–4, Toulouse, France

June 10–11, Washington, D.C.

June 17–18, Chicago, Illinois

Info: www.wolfram.com/services/education/calendar.cgi

June 2–3, 2004

Maîtrise de *Mathematica*

Ritme Informatique

Paris, France

Info: www.ritme.com/fr/formations/programm.html

June 3–4, 2004

M101: A First Course in *Mathematica*

mathemas ordinate

Berlin, Germany

Info: www.wolfram.com/services/education/calendar.cgi

June 4, 2004

Perfectionnement de *Mathematica*

Ritme Informatique

Paris, France

Info: www.ritme.com/fr/formations/programm.html

June 8, 2004

Introduction to *Mathematica*

CANdienst

Amsterdam, Netherlands

Info: www.candienst.nl/english/cursussen/cursussendetail.asp?id=8

June 9, 2004

M100: An Introduction to *Mathematica*

Glass Bead Consulting

London, United Kingdom

Info: www.wolfram.com/services/education/calendar.cgi

June 10–11, 2004

M101: A First Course in *Mathematica*

Glass Bead Consulting

London, United Kingdom

Info: www.wolfram.com/services/education/calendar.cgi

June 14, 16, and 18, 2004

M220: Programming in *Mathematica* Online Class

Wolfram Education Group

Info: www.wolfram.com/services/education/calendar.cgi

June 21, 23, and 25, 2004

M101: A First Course in *Mathematica* Online Class

Wolfram Education Group

Info: www.wolfram.com/services/education/calendar.cgi

July

July, 2004

M110: Working with Data Using *Mathematica*

Wolfram Education Group

July 14, San Francisco, California

Info: www.wolfram.com/services/education/calendar.cgi

M220: Programming in *Mathematica*

Wolfram Education Group

July 15–16, San Francisco, California

July 15–16, Chicago, Illinois

Info: www.wolfram.com/services/education/calendar.cgi

M310: Digital Image Processing

Wolfram Education Group

July 12–13, San Francisco, California

July 15–16, New York, New York

Info: www.wolfram.com/services/education/calendar.cgi

July 9, 2004

Meet UnRisk

uni software plus GmbH

London, United Kingdom

Info: www.unriskderivatives.com/news/index.html

July 10, 17, and 24, 2004

M101: A First Course in *Mathematica* Online Class

Wolfram Education Group

Info: www.wolfram.com/services/education/calendar.cgi

July 12, 2004

Rocky Mountain *Mathematica* 2004

Ed Packel and Stan Wagon

Frisco, Colorado

Info: math.lfc.edu/Rocky_Mtn_Mathematica

July 21, 2004

M110: Working with Data Using *Mathematica*

Glass Bead Consulting

London, United Kingdom

Info: www.wolfram.com/services/education/calendar.cgi

July 22–23, 2004

M101: A First Course in *Mathematica*

Glass Bead Consulting

London, United Kingdom

Info: www.wolfram.com/services/education/calendar.cgi

August

August, 2004

M101: A First Course in *Mathematica*

Wolfram Education Group

August 9–10, San Francisco, California

Info: www.wolfram.com/services/education/calendar.cgi

M330: Neural Networks

Wolfram Education Group

August 12–13, San Francisco, California

August 16–17, Washington, D.C.

Info: www.wolfram.com/services/education/calendar.cgi

M220: Programming in *Mathematica*

Wolfram Education Group

August 19–20, Washington, D.C.

Info: www.wolfram.com/services/education/calendar.cgi

August 1–6, 2004

Sixth International *Mathematica* Symposium

2004 Organizing Committee

Banff, Alberta, Canada

Info: www.ims2004.com

August 5–6, 2004

M101: A First Course in *Mathematica*

mathemas ordinate

Berlin, Germany

Info: www.wolfram.com/services/education/calendar.cgi

August 12–13, 2004

M101: A First Course in *Mathematica*

Glass Bead Consulting

London, United Kingdom

Info: www.wolfram.com/services/education/calendar.cgi

August 18–19, 2004

M101: A First Course in *Mathematica*

Mathematica Nordic Education Centre

Aarhus, Denmark

Info: www.wolfram.com/services/education/calendar.cgi

August 30, September 1 and 3, 2004

M101: A First Course in *Mathematica* Online Class

Wolfram Education Group

Info: www.wolfram.com/services/education/calendar.cgi

September

September, 2004

M110: Working with Data Using *Mathematica* Online Class

Wolfram Education Group

September 7–8

Info: www.wolfram.com/services/education/calendar.cgi

M110: Working with Data Using *Mathematica*

Wolfram Education Group

September 22, New York, New York

Info: www.wolfram.com/services/education/calendar.cgi

M101: A First Course in *Mathematica*

Wolfram Education Group

September 13–14, Seattle, Washington

September 20–21, Las Vegas, Nevada

September 20–21, New York, New York

Info: www.wolfram.com/services/education/calendar.cgi

M220: Programming in *Mathematica*

Wolfram Education Group

September 23–24, New York, New York

Info: www.wolfram.com/services/education/calendar.cgi

September 20–21, 2004

M220: Programming in *Mathematica*

Mathematica Nordic Education Centre

Aarhus, Denmark

Info: www.wolfram.com/services/education/calendar.cgi

September 22–23, 2004

M101: A First Course in *Mathematica*

Mathematica Nordic Education Centre

Copenhagen, Denmark

Info: www.wolfram.com/services/education/calendar.cgi

October

October, 2004

M101: A First Course in *Mathematica*

Wolfram Education Group

October 4–5, San Francisco, California

October 4–5, Washington, D.C.

October 18–19, Boston, Massachusetts

October 25–26, Houston, Texas

October 28–29, New Orleans, Louisiana

Info: www.wolfram.com/services/education/calendar.cgi

M110: Working with Data Using *Mathematica*

Wolfram Education Group

October 6, San Francisco, California

October 6, Washington, D.C.

Info: www.wolfram.com/services/education/calendar.cgi

M220: Programming in *Mathematica*

Wolfram Education Group

October 7–8, San Francisco, California

October 7–8, Washington, D.C.

Info: www.wolfram.com/services/education/calendar.cgi

October 5, 2004

Computeralgebrasimulation mit *Mathematica*

ADDITIVE GmbH

Köln, Germany

Info: www.additive-net.de/software/workshops/sem_koeln.phtml

October 5–6, 2004

M101: A First Course in *Mathematica*

Mathematica Nordic Education Centre

Aarhus, Denmark

Info: www.wolfram.com/services/education/calendar.cgi

October 6, 2004

Computeralgebrasimulation mit *Mathematica*

ADDITIVE GmbH

Heidelberg, Germany

Info: www.additive-net.de/software/workshops/sem_heidelberg.phtml

October 7, 2004

Computeralgebrasimulation mit *Mathematica*

ADDITIVE GmbH

Stuttgart, Germany

Info: www.additive-net.de/software/workshops/sem_stuttgart.phtml

October 7–8, 2004

M101: A First Course in *Mathematica*

mathemas ordinate

Berlin, Germany

Info: www.wolfram.com/services/education/calendar.cgi

October 21–23, 2004

Wolfram Technology Conference

Wolfram Research

Champaign, Illinois

Info: www.wolfram.com/news/events/techconf2004

November

November, 2004

M101: A First Course in *Mathematica*

Wolfram Education Group

November 4–5, Chicago, Illinois

November 4–5, Los Angeles, California

November 15–16, Orlando, Florida

Info: www.wolfram.com/services/education/calendar.cgi

November 8, 10, and 12, 2004

M220: Programming in *Mathematica* Online Class

Wolfram Education Group

Info: www.wolfram.com/services/education/calendar.cgi

November 10–11, 2004

M101: A First Course in *Mathematica*

Mathematica Nordic Education Centre

Aarhus, Denmark

Info: www.wolfram.com/services/education/calendar.cgi

November 15, 17, and 19, 2004

M101: A First Course in *Mathematica* Online Class

Wolfram Education Group

Info: www.wolfram.com/services/education/calendar.cgi

November 15–16, 2004

M220: Programming in *Mathematica*

Mathematica Nordic Education Centre

Copenhagen, Denmark

Info: www.wolfram.com/services/education/calendar.cgi

November 23, 2004

Computeralgebrasimulation mit *Mathematica*

ADDITIVE GmbH

Hannover, Germany

Info: www.additive-net.de/software/workshops/sem_hanover.phtml

November 24, 2004

Computeralgebrasimulation mit *Mathematica*

ADDITIVE GmbH

Köln, Germany

Info: www.additive-net.de/software/workshops/sem_koeln_24_11.phtml

November 25, 2004

Computeralgebrasimulation mit *Mathematica*

ADDITIVE GmbH

Stuttgart, Germany

Info: www.additive-net.de/software/workshops/sem_stuttgart_25_11.phtml

December

December 6–7, 2004

M101: A First Course in *Mathematica*

Mathematica Nordic Education Centre

Copenhagen, Denmark

Info: www.wolfram.com/services/education/calendar.cgi

December 9–10, 2004

M101: A First Course in *Mathematica*

Wolfram Education Group

Atlanta, Georgia

Info: www.wolfram.com/services/education/calendar.cgi

News Bulletins

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

IMS 2004

The Sixth International *Mathematica* Symposium (IMS 2004) is being held August 2 to 6, 2004, in Banff, Canada. For more information, visit www.ims2004.com.

Mathematica and *CalculationCenter* Free Trial Downloads

Now you can download fully functional, 15-day, save-disabled trial versions of *Mathematica* 5 and *CalculationCenter* 2 for free! To order, go to www.wolfram.com/mathematica/trial.cgi or www.wolfram.com/calculationcenter/trial.cgi.

Mathematica Adopted as Exclusive Curriculum Tool in India High Schools

The State Government of Gujarat, one of the largest and most prosperous states in India, passed a resolution to make *Mathematica* a regular part of its math and science curriculum. After an exhaustive search, *Mathematica* was chosen for its power and versatility. To learn more, go to www.wolfram.com/news/gujarat.html.



WOLFRAM 04
TECHNOLOGY CONFERENCE

Learn firsthand about the latest directions in technical computing and see how to incorporate them into your work.

October 21-23, 2004 • Champaign, Illinois, USA
www.wolfram.com/techconf2004

Wolfram Research and UNI-C Now Offer *Mathematica* Training in Nordic Countries

The Nordic Education Centre (NE-C)—a joint venture between Wolfram Research and the Danish IT Centre for Education and Research (UNI-C)—brings certified *Mathematica* training to Denmark, Norway, Sweden, Finland, and Iceland for the first time. For more information, visit www.wolfram.com/news/unic.html.

Computer Science Professor Sculpts Award-Winning Art with *Mathematica*

George Hart has been using *Mathematica* since 1998 to visualize his geometric sculptures, which have appeared in museums, a library, and other public spaces. To learn more, go to <http://www.wolfram.com/news/hart.html>.

Math Professor Uses *Mathematica* to Create Educational Animations

Selwyn Hollis's new book, *A Mathematica Companion for Differential Equations*, shows students how to use *Mathematica* to solve and visualize common differential equations. The book's companion website demonstrates how animations can illustrate practical applications of this facet of calculus. For more information, visit www.wolfram.com/news/hollis.html.

Wolfram Education Group Now Offers Convenient, Online *Mathematica* Training

Wolfram Education Group is now offering certified *Mathematica* training online. Taking an online class requires a phone line and internet connection for web conferencing. For more information, see www.wolfram.com/news/wegonline.html.

UN Branch to Educate Students in Arab States with *Mathematica*

The United Nations Educational, Scientific, and Cultural Organization (UNESCO) has chosen *Mathematica* to improve programs in science and technology in the Arab states. The UNESCO Cairo Office will establish a *Mathematica* demonstration and training center for the 17 Arab states they cover. To learn more, visit www.wolfram.com/news/unesco.html.

***Mathematica* Simulates the Sound of the Big Bang**

Physicist, professor, and science columnist John Cramer used *Mathematica* to produce a 100-second simulation of the sound of the Big Bang. For more information, go to www.wolfram.com/news/bigbangsound.html.

Blind Optical Physicist Pursues Technical Career with the Aid of *Mathematica*

Chuck Strickland, a blind optical physicist with the United States space program, uses *Mathematica* to develop and test algorithms that are used to design optical systems. He created a *Mathematica* program to convert Braille into rich text and is now using *Mathematica* to simulate theoretical models for space telescopes. For more information, go to www.wolfram.com/news/strickland.html.

***Mathematica* 5 for AMD64 Increases Speed by up to 50%**

Mathematica 5 is among the first technical computing platforms specifically optimized for the AMD64 architecture. The new *Mathematica* port outperforms a regular Linux version of *Mathematica* on AMD64 systems by up to 50% in typical scientific and technical calculations. For more information, go to www.wolfram.com/news/amd64.html.

***Mathematica* Helps 17-Year-Old in "Junior Nobel Prize" Contest**

As a senior at Stuyvesant High School, 17-year-old Varun Narendra placed among 40 finalists in the 2003 Intel Science Talent Search, a competition often referred to as the "junior Nobel Prize." Narendra used *Mathematica* to create a model that could help treat Gaucher's disease, a genetic disorder. For more information, go to www.wolfram.com/news/narendra.html.

New Products

***Mathematica* for Students: Semester Edition and Annual Edition**

Innovative Semester and Annual Editions Make Mathematica for Students Accessible for All

Wolfram Research continues to take the lead in responding to the needs of students and faculty with the introduction of the new semester and annual editions of *Mathematica for Students*. These time-limited (six-month and one-year), downloadable editions of *Mathematica for Students* provide an immediate and affordable way for students anywhere in the world to gain the *Mathematica* advantage in any class—for less than half the price of many textbooks.

Mathematica for Students is the technical software preferred by top universities, leading corporations, and government organizations worldwide. With all of the functionality of the professional version, *Mathematica for Students* lets students finish homework assignments and lab reports in record time and is capable of performing all of the computations that students encounter in school from elementary to postgraduate level.

These new downloadable editions will appeal to a broader range of students, including those with less technical majors, who may need the capabilities of *Mathematica* for only one or two prerequisite or elective classes. These editions are also of benefit to faculty who would like to encourage their students to use technical software for their assignments but who previously felt that the added financial burden was too large of a deterrent.

“For a long time now, we have worked with schools and universities to make *Mathematica for Students* as affordable and accessible as possible, offering site purchasing and leasing options that provide added savings over the regular retail price,” states Cliff Hastings, North American Sales Manager at Wolfram Research. “These programs have been widely adopted by many schools, but with the addition of *Mathematica for Students: Semester Edition* and *Mathematica for Students: Annual Edition*, students can now get these savings directly, regardless of whether they are on a licensed *Mathematica* campus.”

“In just 10 minutes, for about \$50, any student around the world can be up and running with *Mathematica*. This opens access to almost all students taking a technical class—even if they aren’t math or science majors,” adds Conrad Wolfram, Director of Strategic International Development. “We’ve removed key barriers to students and their professors opting for *Mathematica*’s unique ability to enhance technical education.”

More info

www.wolfram.com/products/student/mathforstudents/licenses.html

webMathematica 2.1

webMathematica now includes the high-performance *Mathematica* 5, adding more speed and options for real-time, interactive web applications. In addition to introducing a host of innovative features, *Mathematica* 5 delivers large performance gains in dense numerical linear algebra, support for sparse numerical linear algebra, large-scale linear programming, arbitrary-precision computation, and interprocess communication speed.

In addition, webMathematica 2.1 is enhanced with powerful new development options and new graphic, web, and matrix file formats. These new formats—including the popular PNG web bitmap format, the vector graphic format SVG, and the medical DICOM standard—give developers more choices in creating their webMathematica applications.

“We chose webMathematica for math rendering because it has an unrivaled feature set,” says Brandon Lee, CEO of Terra Dotta, a company that provides learning solutions for schools, universities, and businesses. “It does what no other math product can do.”

While the new functionality of webMathematica 2.1 offers authors more options for building applications, the *Mathematica* 5 speed improvements ensure quick delivery of these applications to users. Support for the latest 64-bit platforms and servers allows substantially faster processing of computationally intensive webMathematica applications.

Designed to help developers deliver fast, dynamic websites for their users, webMathematica solves 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 2.1 seamlessly integrates the advances of *Mathematica* 5 into virtually any infrastructure.

More info

www.wolfram.com/products/webmathematica

gridMathematica 1.1

gridMathematica 1.1: Grid Computing Gets a Speed Boost from Mathematica 5

Wolfram Research has significantly increased the performance of its grid computing application with the release of gridMathematica 1.1. gridMathematica now includes *Mathematica* 5, the new high-performance version of Wolfram’s leading technical computing software. The update delivers large performance gains in dense numerical linear algebra, support for sparse numerical linear algebra, large-scale linear programming, arbitrary precision computation, and interprocess communication speed.

Originally unveiled at the Supercomputing 2002 conference, *gridMathematica* is a complete parallel computing solution for dedicated grids or clusters. It offers a highly cost-effective way of deploying *Mathematica* for parallel computations. *gridMathematica* can run on any cluster of machines, including Unix, Linux, Windows, and Mac OS X, and requires only TCP/IP connectivity. Version 1.1 capitalizes on a new TCP/IP protocol in *Mathematica 5* that allows it to communicate at the speed of the underlying network. On most standard networks this means that users will see bandwidth improvements of a factor of 10 and latency improvements of a factor of 200; on faster networks the gains are even higher.

With *gridMathematica*, simple commands such as `ParallelEvaluate`, `ParallelTable`, and `ParallelMap` can often be used to quickly parallelize sequential codes or develop parallel applications from scratch. *gridMathematica* supports all common parallel computing constructs such as virtual shared memory, distributed memory, automatic or explicit process scheduling, explicit concurrency primitives, and failure recovery of stranded processes.

“*Mathematica 5* now offers the fastest technical computing on a single computer. *gridMathematica 1.1* multiplies that by the grid to provide the best high-performance computing environment available,” says Roger Germundsson, Director of R & D at Wolfram Research. Both individual users and large-scale organizations can optimize their computations with this sophisticated, easy-to-use alternative for high-level programming applications.

More info

www.wolfram.com/products/gridmathematica

CalculationCenter 2 for Networks and Students

Wolfram Research has expanded its licensing options for *CalculationCenter*, making the easy-to-use computational software even more accessible. The release of *Network CalculationCenter* and *CalculationCenter for Students* streamlines affordable multi-user site access and offers students an inexpensive option for use on their personal computers.

Based on *Mathematica* technology, but specifically designed for less demanding practical applications, *CalculationCenter*'s new Version 2 features quick data analysis, enhanced error-correction capabilities, and broadened communications support. With a simple and friendly interface, *CalculationCenter 2* eliminates the need for lengthy manuals and lets users get right to the tasks at hand.

Network CalculationCenter is the most cost-effective licensing solution for academic and business settings where dedicated single-machine licenses are not necessary. With this new option, sites can optimize the availability of a given number of licenses over a larger group of users.

Using the *MathLM* licensing manager, system administrators can deploy multiple processes from a central location with just one installation and offer their users the convenience of working from any networked computer.

While the new network licensing provides more flexible options to multiple-user sites, *CalculationCenter for Students* offers more affordability for students working at their personal computers.

With immediate support for anything from basic algebra to symbolic integration and differentiation, *CalculationCenter for Students* is ideal for students enrolled in business, statistics, and other related courses. Integrated help systems like Instant-Calculators assist in getting assignments done quickly and correctly on the first try.

Using the SmartPlot feature, students may also create complex graphs to be exported in 3D file formats, integrated into technical word-processing documents compatible with notebooks (.nb), or published on the web. With *CalculationCenter for Students*, professional-quality presentations are only a few clicks away.

Now available with licensing options to suit any user's needs, *CalculationCenter 2* is the solution for technical workloads of almost any level and size.

"*CalculationCenter* is already an indispensable tool for many students, professionals, and educators," said Conrad Wolfram, Director of Strategic and International Development at Wolfram Research. "With these new competitively priced editions, *CalculationCenter* will be more accessible for training toward technical careers and for organizations looking to integrate the quick computational solution into their R&D infrastructure."

More info

www.wolfram.com/products/calculationcenter

New *Mathematica* Applications and Upgrades

GUIKit 1.0 Beta

GUIKit simplifies the construction and layout of common graphical user interface (GUI) programming components while minimizing the need to know any details of the underlying Java framework.

More info

www.wolfram.com/solutions/mathlink/guikit

Magnetica

This new tool lets you analyze, design, optimize, and simulate magnetic systems with *Mathematica*.

More info

www.wolfram.com/news/magnetica.html

MathOptimizer Professional

MathOptimizer Professional is a recent application that combines the power of *Mathematica* with the established LGO (Lipschitz Global Optimizer) solver suite. It offers sophisticated application development tools and enables the global and local solution of a general class of continuous optimization problems.

More info

www.wolfram.com/products/applications/mathoptpro

SchematicSolver 2

SchematicSolver 2 offers users working in signal processing, systems, and circuits enhanced functionality unavailable in any other software for drawing and solving schematic systems.

More info

www.wolfram.com/news/schematicsolver2.html

Global Optimization 4.3

Global Optimization 4.3 is a collection of functions for constrained and unconstrained global nonlinear optimization. Any function computable by *Mathematica* can be used as input.

More info

www.wolfram.com/products/applications/globalopt

MathCode C++ 1.2

With *MathCode C++* you can generate optimized C++ code that can be compiled and connected seamlessly into *Mathematica*.

More info

www.wolfram.com/products/applications/mathcode

mathStatICA 1.5

Built upon the incredible symbolic computational power of *Mathematica*, *mathStatICA* was designed to find the exact symbolic solutions to statistical problems.

More info

www.wolfram.com/products/applications/mathstatICA

TSi ProPac 3.2

TSi ProPac is an integrated, enhanced version of *TSi Dynamics* and *TSi Controls*. This package enables users to build plant models, create simulations, obtain linear models, design linear and nonlinear controls, design adaptive controls, and implement controllers.

More info

www.wolfram.com/products/applications/tsipropac

New Resources

The Wolfram Functions Site

The Wolfram Functions Site is a compendium of 87,160 formulas and 10,828 graphics about mathematical functions. The Wolfram Functions Site assembles all of these and makes them immediately accessible for human or computer use. This free resource is available at functions.wolfram.com.

Wolfram Teacher Network

The Wolfram Teacher Network is a new resource for sharing knowledge among science and math teachers. Teachers submit questions about classroom applications for *Mathematica*, and Wolfram Research matches them with the appropriate *Mathematica* mentors, who are experienced instructors eager to help. For more information, visit www.wolfram.com/precollege/network.

A New Kind of Science Online

Stephen Wolfram's groundbreaking book *A New Kind of Science* is now freely available online, with complete text and images, over 30,000 links, and many other enhanced features. NKS|Online has been expanded to include the preface notes and bibliography papers, with more enhancements planned. To access the complete work, visit www.wolframscience.com/nksonline.

Selected Talks by Stephen Wolfram Available Online

These transcripts include the full text, images, and animations as presented to the original audiences. The site has been updated to include Wolfram's most recent talks, including an invited address at the Joint Mathematics Meeting and a videoconference talk at the International Symposium on Modern Computing. To access these presentations, visit www.stephenwolfram.com/publications/talks.

***Mathematica* Information Center**

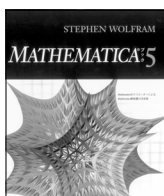
A number of new notebooks and packages have been added.

- The 536 distinct solutions to the world's oldest puzzle, the Loculus of Archimedes, can be downloaded at library.wolfram.com/infocenter/MathSource/5108.
- A new coding theory package is available for download at library.wolfram.com/infocenter/MathSource/5085.
- AELink, a *Mathematica* application that allows Mac OS X users to send arbitrary Apple events from within *Mathematica*, can be downloaded at library.wolfram.com/infocenter/MathSource/5114.
- Construct 3D zonotiles with the new *Mathematica* notebook available for download at library.wolfram.com/infocenter/MathSource/1197.
- Eric Weisstein's *MathWorld* packages can be downloaded at library.wolfram.com/infocenter/MathSource/4775.

Keep up to date with the new items added to the *Mathematica* Information Center using the Wolfram Research RSS feed. To start using this RSS feed, just enter the URL below into your RSS reader:

library.wolfram.com/infocenter/WhatsNew/?content=rss

New Publications



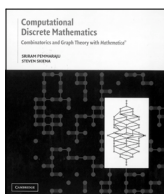
The *Mathematica* Book, Fifth Edition (Japanese translation)

Stephen Wolfram, 2003, Wolfram Media, Inc.,
1,424 pp., hardcover

Japanese translation of *The Mathematica Book*.

Available:

store.wolfram.com/view/book/ISBN1579550231.str



Computational Discrete Mathematics: Combinatorics and Graph Theory with *Mathematica*

Sriram Pemmaraju and Steven Skiena, 2003, Cambridge
University Press, 480 pp., hardcover

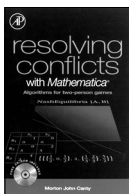
This book is a reference and user's guide for *Combinatorica*, an extension to *Mathematica* that is used for teaching and research in discrete mathematics. Included are examples of all 450 *Combinatorica* functions as well as associated mathematical and algorithmic theory. These functions are available for active experimentation and visualization with the aim of advancing the study of combinatorics and graph theory. The text covers classical and advanced topics on the most important combinatorial objects and all important areas of graph theory, and can be used for self-study or in discrete mathematics courses.

Chapters

Combinatorica: An Explorer's Guide | Permutations and Combinations | Algebraic Combinatorics | Partitions, Compositions, and Young Tableaux | Graph Representation | Generating Graphs | Properties of Graphs | Algorithmic Graph Theory

Available:

store.wolfram.com/view/book/ISBN0521806860.str



Resolving Conflicts with *Mathematica*: Algorithms for Two-Person Games

Morton John Canty, 2003, Academic Press, 327 pp., hardcover

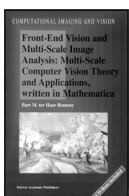
This revised edition of a popular German textbook provides an accessible introduction to game theory and can be used as a study guide for university students or as a tool for practitioners in game-theoretical modeling. The application of algorithmic game theory to resolve realistic conflict situations is emphasized, and *Mathematica* is used to obtain both numerical and analytical solutions to game-theoretical models. The accompanying CD-ROM includes notebooks, programs, and exercises with their solutions for each chapter, and topics covered in the text include extensive and normal form games, degeneracy, and equilibrium selection theory.

Chapters

Non-Cooperative Games | Linear Complementarity | Zero-Sum Games | Degenerate Games | Inspection Games | Evolutionary Games | Games in Extensive Form

Available:

store.wolfram.com/view/book/ISBN0121588556.str



Front-End Vision and Multi-Scale Image Analysis

Bart M. ter Haar Romeny, 2003, Kluwer Academic Publishers, 464 pp., paperback

Intended for undergraduate and graduate students and all who are interested in computer vision, medical imaging, and human visual perception, this interactive book presents a tutorial approach to mastering the mathematics of computer vision and image analysis. One purpose of the book is to bridge the gap between the world of human visual perception and the world of multi-scale computer vision theory and applications. All chapters are written in *Mathematica*, and the text and included code invite readers to explore computer vision on their own.

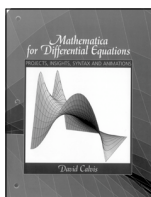
Chapters

Front-End Vision and Multi-scale Image Analysis | Apertures and the Notion of Scale | Foundations of Scale-Space | The Gaussian Kernel | Gaussian Derivatives | Multi-scale Derivatives: Implementations | Differential Structure of Images | Natural Limits on Observations | Differentiation and Regularization |

The Front-End Visual System—the Retina | A Scale-Space Model for the Retinal Sampling | The Front-End Visual System—LGN and Cortex | The Front-End Visual System—Cortical Columns | Deep Structure I. Watershed Segmentation | Deep Structure II. Catastrophe Theory | Deep Structure III. Topological Numbers | Deblurring Gaussian Blur | Multi-scale Optic Flow | Color Differential Structure | Steerable Kernels | Scale-Time | Geometry-Driven Diffusion | Epilog: Introduction to *Mathematica*, The Concept of Convolution, Installing the Book and Packages, First Start with *Mathematica*: Tips & Tricks

Available:

store.wolfram.com/view/book/ISBN1402015070.str



***Mathematica* for Differential Equations: Projects, Insights, Syntax and Animations**

David Calvis, 2004, Pearson Prentice Hall, 160 pp., paperback

This manual draws on the capabilities of *Mathematica* to enrich the subject of differential equations and boundary value problems and can be used with any introductory differential equations course. The manual includes a wide variety of stand-alone worksheets that lead students into insights that would be difficult to obtain without the computer. No prior knowledge of *Mathematica* is assumed, but students learn the basics of *Mathematica* syntax as they gain experience with the assignments.

Chapters

First-Order Differential Equations | Mathematical Models and Numerical Methods | Linear Equations of Higher Order | Introduction to Systems of Differential Equations | Linear Systems of Differential Equations | Nonlinear Systems and Phenomena | Laplace Transform Methods | Power Series Methods | Fourier Series Methods | Eigenvalues and Boundary Value Problems

Available:

store.wolfram.com/view/book/ISBN0131439766.str



Mathematica Computer Guide

E. Kreyszig and E. J. Norminton, John Wiley & Sons,
257 pp., paperback

This self-contained introductory guide was written in *Mathematica* 4 to supplement *Advanced Engineering Mathematics*, Eighth Edition, by Erwin Kreyszig. The guide presents *Mathematica* as simply as possible and includes over 130 worked-out examples and about 400 problems for the student to solve. Answers to the odd-numbered problems are included in an appendix to the guide.

Chapters

Introduction, General Commands

Part A: Ordinary Differential Equations (ODEs)

First-Order ODEs | Linear ODEs of Second and Higher Order | Systems of Differential Equations, Phase Plane, Qualitative Methods | Series Solutions of Differential Equations | Laplace Transform Method for Solving ODEs

Part B: Linear Algebra, Vector Calculus

Matrices, Vectors, Determinants, Linear Systems of Equations | Matrix Eigenvalue Problems | Vectors in R^2 and R^3 , Dot and Cross Products, Grad, Div, Curl | Vector Integral Calculus, Integral Theorems

Part C: Fourier Analysis and Partial Differential Equations

Fourier Series, Integrals, and Transforms | Partial Differential Equations (PDEs)

Part D: Complex Analysis

Complex Numbers and Functions, Conformal Mapping | Complex Integration | Power Series, Taylor Series | Laurent Series, Residue Integration | Complex Analysis in Potential Theory

Part E: Numerical Methods

Numerical Methods in General | Numerical Linear Algebra | Numerical Methods for Differential Equations

Part F: Optimization, Graphs

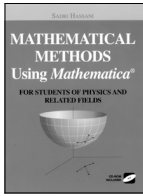
Unconstrained Optimization, Linear Programming

Part G: Probability and Statistics

Data Analysis, Probability Theory | Mathematical Statistics

Available:

store.wolfram.com/view/book/ISBN0471386693.str



Mathematical Methods Using *Mathematica*: For Students of Physics and Related Fields

Sadri Hassani, 2003, Springer-Verlag, 235 pp., paperback

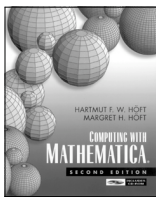
A self-contained introduction to the use of computer algebra for undergraduates in the physical sciences or engineering, this book teaches the essentials of *Mathematica* through examples from undergraduate physics and discusses the theory of numerical integration, discrete differentiation, and the solution of differential equations. The book is intended for use with any standard textbook in mathematical methods for science and engineering. A large number of numerical topics and exercises are presented, and the included CD-ROM contains *Mathematica* notebooks that illustrate the topics in the text as well as solve problems in mathematical physics.

Chapters

Mathematica in a Nutshell | Vectors and Matrices in *Mathematica* | Integration | Infinite Series and Finite Sums | Numerical Solutions of ODEs: Theory | Numerical Solutions of ODEs: Examples Using *Mathematica*

Available:

store.wolfram.com/view/book/ISBN0387955232.str



Computing with *Mathematica*, Second Edition

Hartmut F. Höft and Margret H. Höft, 2003, Academic Press, 313 pp., paperback

Designed for the junior/senior level of the mathematics and computer science curricula, this book teaches students how to use *Mathematica* to solve problems in mathematics, computer science, engineering, and other scientific fields in which mathematics and computing skills are essential. The text moves from simple to complex and often follows a specific real-world example on a number of different levels, allowing students to build their confidence and competence. The accompanying CD-ROM contains the entire text of the book as a collection of *Mathematica* notebooks.

Chapters

Part I: Learning the Basics

Mathematica Basics: An Introduction | *Mathematica* Basics: Help | *Mathematica* Basics: Notebooks | *Mathematica* Basics: Text and Typesetting | *Mathematica* Basics: Packages

Part II: Designing Functions

Values, Variables, and Assignments | Functions | Recursive Definitions | Substitution Rules and Optional Arguments | Four Spheres Packing Problem

Part III: Designing Programs

List Processing Functions | Iterations with Loops | Computations with Modules and Local Variables

Part IV: Exploring Advanced Features

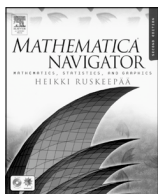
Advanced *Mathematica*: Options | Advanced *Mathematica*: Hyperlinks and Buttons | Advanced *Mathematica*: Packages | Advanced *Mathematica*: Files, Data Exchange, and Conversions

Part V: Student Projects

Student Projects

Available:

store.wolfram.com/view/book/ISBN0123516668.str

***Mathematica Navigator: Mathematics, Statistics, and Graphics, Second Edition***

Heikki Ruskeepää, 2004, Elsevier Academic Press, 844 pp., paperback

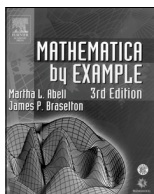
Designed for students with no previous knowledge of *Mathematica*, this book emphasizes applied mathematical and statistical methods, graphics, and programming. It will take readers through complex mathematical topics with an emphasis on graphic representation. This new edition incorporates thorough revisions, extensive information about *Mathematica* as a writing tool, and is now based on *Mathematica* 5.

Chapters

Preface | Starting | Sightseeing | Notebooks | Files | 2D Graphics for Functions | Options for 2D Graphics | 3D Graphics for Functions | Options for 3D Graphics | 2D Graphics for Data | 3D Graphics for Data | Numbers | Expressions | Lists | Functions | Programs | Differential Calculus | Integral Calculus | Matrices | Equations | Optimization | Interpolations | Approximation | Differential Equations | Partial Differential Equations | Difference Equations | Probability | Statistics | References | Index

Available:

store.wolfram.com/view/book/ISBN012603642X.str



***Mathematica* by Example, Third Edition**

Martha L. Abell and James P. Braselton, 2004, Elsevier Academic Press, 571 pp., paperback

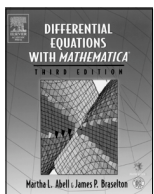
This *Mathematica* reference for beginners is suitable for students, instructors, engineers, business people, and other professionals who are just learning to use *Mathematica*. The text has been updated to be completely compatible with *Mathematica* 5.0 and includes new applications from the fields of biology, physics, and engineering. The basic commands as well as commands that are useful in areas such as calculus, linear algebra, business mathematics, ordinary and partial differential equations, and graphics are introduced, and all commands are followed by examples. The accompanying CD-ROM includes the *Mathematica* input that appears in the book.

Chapters

Getting Started | Basic Operations on Numbers, Expressions, and Functions | Calculus | Introduction to Lists and Tables | Matrices and Vectors | Differential Equations

Available:

store.wolfram.com/view/book/ISBN0120415631.str



Differential Equations with *Mathematica*, Third Edition

Martha L. Abell and James P. Braselton, 2004, Elsevier Academic Press, 876 pp., paperback

This new edition introduces students, instructors, and scientists to solving differential equations with *Mathematica* 5. It first covers topics typical in undergraduate calculus courses and then illustrates how *Mathematica* enhances these lessons by overcoming visual and computational difficulties.

Chapters

Preface | Introduction to Differential Equations | First-Order Ordinary Differential Equations | Applications of First-Order Ordinary Differential Equations | Higher-Order Differential Equations | Applications of Higher-Order Differential Equations | Systems of Ordinary Differential Equations | Applications of Systems of Ordinary Differential Equations | Laplace Transform Methods | Eigenvalue Problems and Fourier Series | Partial Differential Equations | Appendix: Getting Started | The *Mathematica* Menu | Bibliography | Index

Available:

store.wolfram.com/view/book/ISBN0120415623.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* news group, `comp.soft-sys.math.mathematica`.

■ Symbolic Analysis of DNA Sequences

Jose Manuel Gutiérrez

gutierjm@unican.es

DNA chains can be viewed as symbolic sequences of four possible nucleotides—adenine (A), cytosine (C), thymine (T), and guanine (G)—that define the structure of amino acids to form proteins. DNA sequences are available at the GenBank[®] database of the National Center for Biotechnology Information (NCBI) at www.ncbi.nlm.nih.gov/entrez/viewer.fcgi as plain ASCII files that contain information about the organism, as well as the sequence of nucleotides. Discovering the structure encoded in DNA sequences is of paramount importance.

Given a symbolic sequence $s_1, s_2, \dots, s_n$, consider the dynamical system $x_{i+1} = f_{s_i}(x_i)$, $i = 1, \dots, n$, which defines an *orbit* on the attractor of the iterated function sequence (IFS) from an arbitrary initial condition x_0 . The orbit only visits those regions of the attractor associated with the words appearing in the symbolic sequence. Jeffrey [1] applied this method to visualize the attractor of the IFS represented by DNA sequences.

The following commands implement an efficient version of the chaos-game algorithm for an IFS consisting of four transformations. We map each symbol, a, b, c, d to a corner of the unit square, $a \rightarrow \{0, 0\}$, $b \rightarrow \{0, 1\}$, $c \rightarrow \{1, 0\}$, $d \rightarrow \{1, 1\}$. **FoldList** is used to compute the orbit and compilation is used to speed up the computation.

```
In[1]:= Orbit[s_List, {a_, b_, c_, d_}] :=  
      OrbitMap[s /. {a -> {0, 0}, b -> {0, 1}, c -> {1, 0}, d -> {1, 1}}]  
  
In[2]:= OrbitMap = Compile[{{m, _Real, 2}}, FoldList[  
       $\frac{\#1 + \#2}{2}$  &, {0, 0}, m]];
```

```

In[3]:= IFSPlot[s_List] :=
  Show[Graphics[{AbsolutePointSize[0.5], Point /@ Orbit[s, Union[s]]}],
    AspectRatio → Automatic, PlotRange → {{0, 1}, {0, 1}},
    GridLines → {Range[0, 1,  $\frac{1}{2^3}$ ], Range[0, 1,  $\frac{1}{2^3}$ ]}]

```

Symbolic sequences can be extremely long so it is important to efficiently load and process such data. Moreover, if the data is available via the internet, there is no need to store it locally. Rolf Mertig supplied code for opening and reading from a URL stream using *J/Link*.

```

In[4]:= Needs["JLink`"];
ImportURL[url_String] := JavaBlock[
  Module[{u, s, stream, numRead, outFile, buf},
    InstallJava[];
    u = JavaNew["java.net.URL", url];
    stream = u[openStream[]];
    If[stream === $Failed, Return[$Failed]];
    buf = JavaNew["B", 5000]; s = "";
    While[(numRead = stream[read[buf]]) > 0, s = s <> FromCharacterCode[
      (If[# < 0, # + 256, #] &) /@ Take[Val[buf], numRead]]];
    stream[close[]];
    s
  ]
];

```

The following code reads in DNA sequences as a string of characters, separately printing out descriptive information.

```

In[6]:= NCBI = "http://www.ncbi.nlm.nih.
gov/entrez/viewer.fcgi?txt=on&list_uids=";

In[7]:= DNASequences[id_] := Module[{lines = Import[StringToStream[
  ImportURL[NCBI <> ToString[id] <> "&view=fasta"]], "Lines"]},
  Print[First[lines]];
  Characters[StringJoin@@Rest[lines]]]

```

There are two types of **sequence identification numbers**: *GI numbers* (a series of digits that are assigned consecutively by NCBI to each sequence it processes) and *version numbers* (which consist of the accession number followed by a dot and a version number). Either format can be used. Now, we use the previous command to import the mouse mitochondrion and man mitochondrion complete genomes.

```

In[8]:= mouse = DNASequences["342520"];

```

```
>gi|342520|gb|J01420.1|MUSMTCG Mouse mitochondrion, complete genome
```

```

In[9]:= man = DNASequences["X93334.1"];

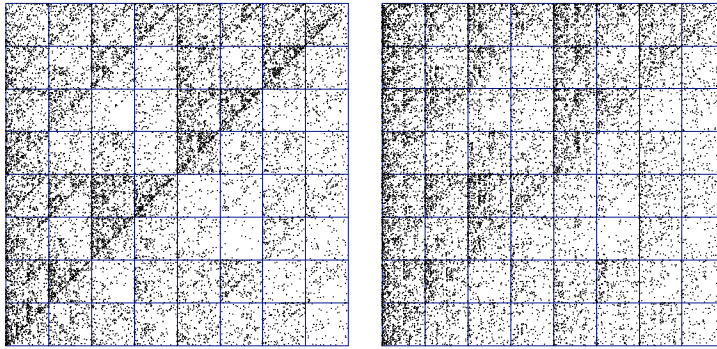
```

```
>gi|1262342|emb|X93334.1|HSMITG
Homo sapiens mitochondrial DNA, complete genome
```

Both genomes define similar patterns (they are close in the phylogenetic tree: mitochondrial eukaryotes, vertebrata). Perhaps it is, after all, hard to answer the question, “Are you a man or a mouse?”

```
In[10]:= Needs["Graphics`"];
         DisplayTogetherArray[IFSPlot /@ {mouse, man}]
```

From In[10]:=



■ Generating Barcodes

We encounter a wide range of barcodes nearly every day. At www.barcodeman.com/info/c39_1.php one can find a number of barcode specifications. In one popular code—Code 39—each character is made up of 9 bars, alternating black and white, 3 of which are wider than the others. A single character therefore consists of 5 black bars and 4 white bars. The ratio of the bar widths, w , can range from 2.2:1 to 3:1 and the total width of a character is $3w + 6$. The *intercharacter gap* (the space between each barcode character) is undefined, but is usually equivalent to a narrow white bar.

The Code 39 barcode symbology supports 43 characters plus *, which is used as a delimiter or start/stop character. Note that the alphabetic characters are all upper case. Here is the bar width configuration table. A zero corresponds to a narrow bar and a one to a wide bar.

" "	0	1	1	0	0	0	1	0	0
"\$"	0	1	0	1	0	1	0	0	0
"%"	0	0	0	1	0	1	0	1	0
"&"	0	1	0	0	1	0	1	0	0
"+"	0	1	0	0	0	1	0	1	0
"_"	0	1	0	0	0	0	1	0	1
","	1	1	0	0	0	0	1	0	0
"/"	0	1	0	1	0	0	0	1	0
"0"	0	0	0	1	1	0	1	0	0
"1"	1	0	0	1	0	0	0	0	1
"2"	0	0	1	1	0	0	0	0	1

```

In[1]:= code39 = {
  "3"  1 0 1 1 0 0 0 0 0
  "4"  0 0 0 1 1 0 0 0 1
  "5"  1 0 0 1 1 0 0 0 0
  "6"  0 0 1 1 1 0 0 0 0
  "7"  0 0 0 1 0 0 1 0 1
  "8"  1 0 0 1 0 0 1 0 0
  "9"  0 0 1 1 0 0 1 0 0
  "A"  1 0 0 0 0 1 0 0 1
  "B"  0 0 1 0 0 1 0 0 1
  "C"  1 0 1 0 0 1 0 0 0
  "D"  0 0 0 0 1 1 0 0 1
  "E"  1 0 0 0 1 1 0 0 0
  "F"  0 0 1 0 1 1 0 0 0
  "G"  0 0 0 0 0 1 1 0 1
  "H"  1 0 0 0 0 1 1 0 0
  "I"  0 0 1 0 0 1 1 0 0
  "J"  0 0 0 0 1 1 1 0 0
  "K"  1 0 0 0 0 0 0 1 1
  "L"  0 0 1 0 0 0 0 1 1
  "M"  1 0 1 0 0 0 0 1 0
  "N"  0 0 0 0 1 0 0 1 1
  "O"  1 0 0 0 1 0 0 1 0
  "P"  0 0 1 0 1 0 0 1 0
  "Q"  0 0 0 0 0 0 1 1 1
  "R"  1 0 0 0 0 0 1 1 0
  "S"  0 0 1 0 0 0 1 1 0
  "T"  0 0 0 0 1 0 1 1 0
  "U"  1 1 0 0 0 0 0 0 1
  "V"  0 1 1 0 0 0 0 0 1
  "W"  1 1 1 0 0 0 0 0 0
  "X"  0 1 0 0 1 0 0 0 1
  "Y"  1 1 0 0 1 0 0 0 0
  "Z"  0 1 1 0 1 0 0 0 0
};

```

Corresponding to a single character, the following code generates a sequence of rectangles of alternating black and white color (via **MapIndexed** and **Mod**) according to the bar width configuration table.

```

In[2]:= BarCode[s_String, pos_, w_] :=
  Module[{code = Rest[code39[Position[code39, ToUpperCase[s]]][1, 1]]] /.
    {0 → 1, 1 → w}},
    Graphics[MapIndexed[{GrayLevel[Mod[First[#2] - 1], 2]],
      Rectangle[{First[#1], 0}, {Last[#1], 1}] &,
      Partition[FoldList[Plus, pos, code], 2, 1], 1]]] /;
  MemberQ[code39[All, 1], ToUpperCase[s]]

```

The position of each successive rectangle is computed using **FoldList** (computing a cumulative sum starting from **pos**). Since the alphabetic characters are all upper case, lowercase input is converted using **ToUpperCase**. The replacement rule $\{0 \rightarrow 1, 1 \rightarrow w\}$ is used to map from zeros and ones to narrow bars (of unit width) and wide bars (of width w), respectively.

Adding 1 for the intercharacter gap, the total width of a character is $3w + 7$. Generating the barcode for a string of characters is straightforward, again using **MapIndexed**. **StringJoin** is used to prepend and append the start/stop character *. The default ratio of widths is 3 (via **w_**: 3).

```

In[3]:= BarCode[word_String, w_: 3] :=
  MapIndexed[BarCode[#1, (3 w + 7) (First[#2] - 1), w] &,
    Characters["*" <> word <> "*"]]

```

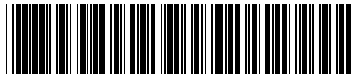
Here is the barcode of *Mathematica* in Code 39.

```

In[4]:= Show[BarCode["Mathematica"], AspectRatio → 1 / 5]

```

From In[4]:=



By adding catch all code for missing characters, we can display such characters as a “graphical error.”

```

In[5]:= BarCode[s_String, pos_, w_] :=
  Graphics[{Hue[1], Rectangle[{pos, 0}, {pos + (3 w + 7), 1}]}]

```

For example, # and ^ are not in Code 39 and are shown in red.

```

In[6]:= Show[BarCode["A#B^C"], AspectRatio → 1 / 5]

```

From In[6]:=



■ Computing Your Tax

A nice application of **Interpolation** is to compute how much tax you have to pay. For example, in Australia the income tax thresholds are as follows.

Income \$	Tax Rate
0 – 6,000	0 %
6,001 – 21,600	17 %
21,601 – 52,000	30 %
52,001 – 62,500	42 %
> 62,501	47 %

Express these thresholds as a matrix (assuming a maximum salary of \$1,000,000).

$$\text{In[1]:= thresholds} = \begin{pmatrix} 0 & 0 \\ 6000 & 0 \\ 21600 & 0.17 \\ 52000 & 0.30 \\ 62500 & 0.42 \\ 10^6 & 0.47 \end{pmatrix};$$

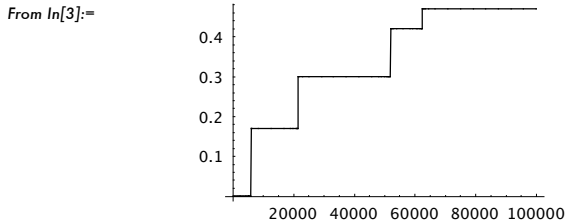
Then use zeroth-order interpolation to construct the tax rate function, **tax'**.

$$\text{In[2]:= tax'} = \text{Interpolation}[\text{thresholds}, \text{InterpolationOrder} \rightarrow 0]$$

$$\text{Out[2]= InterpolatingFunction}[(0. \ 1. \times 10^6), <>]$$

A plot of the tax rate shows that it has the required stepwise behavior.

$$\text{In[3]:= Plot}[\text{tax}'(i), \{i, 0, 10^5\}]$$



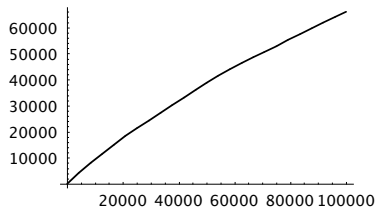
Integrating the tax rate gives the tax as a function of income. Note that indefinite integration using **Integrate** automatically returns an **InterpolatingFunction** over the same domain.

$$\text{In[4]:= tax}(i) = \int \text{tax}'(i) \, di;$$

Here is a plot of the after-tax income versus gross income.

```
In[5]:= Plot[i - tax(i), {i, 0, 105}]
```

From In[5]:=



■ ParentForm

The default input and output **FormatType** is **StandardForm**. An easy way to define a new format is to use **ParentForm**. All rules are inherited from the **ParentForm** and new rules can be added and old rules overridden if desired. Here we extend **StandardForm** to **ExtendedStandardForm**.

There are several acts of “registration” needed to add a new **FormatType** so that it works in input cells in the front end:

1. Define the **ParentForm** of **ExtendedStandardForm** in the kernel.

```
Unprotect[ParentForm];
ParentForm[ExtendedStandardForm] = StandardForm;
Protect[ParentForm];
```

2. Augment **\$BoxForms** in the kernel.

```
$BoxForms = Append[$BoxForms, ExtendedStandardForm];
```

3. Augment the front end’s menu of box forms.

```
SetOptions[$FrontEnd, BoxFormFormatTypes →
  {"StandardForm", "TraditionalForm", "ExtendedStandardForm"}];
```

The new **ExtendedStandardForm** format appears under the appropriate categories of the Cell menu.

4. Add a new cell style for **ExtendedStandardForm**.

Since the **ParentForm** of **ExtendedStandardForm** is **StandardForm**, make a copy of the **StandardForm** cell style definition, rename it **ExtendedStandardForm**, and put it in the *FormatType Styles* section of the notebook’s style sheet. I have modified this cell’s style definition by coloring it red.

```
Prototype for style: "ExtendedStandardForm":
ExtendedStandardForm
```

5. Make **ExtendedStandardForm** your default input and output format via Default Input FormatType and Default Output FormatType under the Cell menu.

Presently **StandardForm** and **ExtendedStandardForm** are identical, apart from the coloring of input and output.

$2 + 2$

4

6. Add or modify formatting rules in **ExtendedStandardForm**.

Suppose that you would like to automatically interpret mappings, $f : x \mapsto y$, as *Mathematica* functions, $f := \text{Function}[x, y]$. To do this, add a formatting rule to **ExtendedStandardForm** using the **Utilities`Notation`** package.

```
<<Utilities`Notation`
```

```
Notation[f_ : x_  $\mapsto$  y_  $\Leftrightarrow$  f_ := Function[x_, y_]]
```

This definition is now part of **ExtendedStandardForm**. We also add an input alias to simplify entering this notation.

```
AddInputAlias[ $\blacksquare$  :  $\square \mapsto \square$ , "map"]
```

Now, if you type $\text{\texttt{ESC}}\text{map}\text{\texttt{ESC}}$ in an input cell you obtain $\blacksquare : \square \mapsto \square$. Using this alias, enter the map $f : x \mapsto x \cos(x^2)$, moving between placeholders via the $\text{\texttt{TAB}}$ key.

```
f : x  $\mapsto$  x Cos[x2]
```

After entering this cell, things work as expected.

```
f[y]
```

```
y Cos[y2]
```

If we duplicate the above cell containing the definition of f and then do **Cell** $\triangleright$ **Convert To** $\triangleright$ **StandardForm**, we obtain the **StandardForm** representation of this cell.

```
f := Function[x, x Cos[x2]]
```

Alternatively, if we enter a function using **Function** notation, say $f := \text{Function}[x, \text{Sin}[x]/x]$, select the cell, and then do **Cell** $\triangleright$ **Convert To** $\triangleright$ **ExtendedStandardForm**, we obtain the **ExtendedStandardForm** representation of this function.

```
f : x  $\mapsto$   $\frac{\text{Sin}[x]}{x}$ 
```

7. Restore the default input and output formats via the **Cell** menu.

8. Restore the front end box forms using **SetOptions**.

```
SetOptions[$FrontEnd,  
BoxFormFormatTypes  $\rightarrow$  {"StandardForm", "TraditionalForm"}];
```

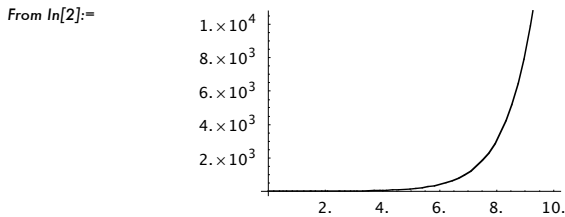
■ ScientificForm in Ticks

Bob Hanlon

bobhanlon@aol.com

Here is a simple way to obtain **ScientificForm** ticks in a graph.

```
In[1]:= << Graphics`
In[2]:= Module[{p1, t}, DisplayTogether[p1 = Plot[E^x, {x, 0, 10}],
  Ticks -> {(t = Ticks /. AbsoluteOptions[p1, Ticks])[1],
    t[[2]] /. {y_, _?NumericQ, r_} -> {y, ScientificForm[y, r}]]]
```



■ Binary Variable Equations

Lars Rasmusson

Lars.Rasmusson@hp.com

Consider the following problem: Johnny's ideal woman is red-haired, green-eyed, slender, and tall. He knows four women.

```
In[1]:= women = {Adele, Betty, Carol, Doris};
```

Here are the requirements that they meet:

1. Only three of the women are both green-eyed and slender.
2. Only two of the women are both red-haired and tall.
3. Only two of the women are both slender and tall.
4. Only one of the women is both green-eyed and red-haired.
5. Adele and Betty have the same color eyes.
6. Betty and Carol have the same color hair.
7. Carol and Doris have different builds.
8. Doris and Adele are the same height.
9. Only one of the four women has all four characteristics.

To determine which one of the four women satisfies all of Johnny's requirements, one can set up and solve a system of equations involving *binary variables*: **redhair[[2]] == 1** if **Betty** is red-haired, and so forth.

Define the binary variables as elements of lists corresponding to each attribute.

```
In[2]:= {redhair, greeneye, slender, tall, perfectgirl} =
        Array[#, Length[women]] & /@ {r, g, s, t, p}
```

$$\text{Out[2]} = \begin{pmatrix} r(1) & r(2) & r(3) & r(4) \\ g(1) & g(2) & g(3) & g(4) \\ s(1) & s(2) & s(3) & s(4) \\ t(1) & t(2) & t(3) & t(4) \\ p(1) & p(2) & p(3) & p(4) \end{pmatrix}$$

There are 20 variables.

```
In[3]:= vars = Flatten[%]
```

```
Out[3]= {r(1), r(2), r(3), r(4), g(1), g(2), g(3), g(4), s(1),
         s(2), s(3), s(4), t(1), t(2), t(3), t(4), p(1), p(2), p(3), p(4)}
```

Use $x^2 = x$ to constrain the variables to be {0, 1}.

```
In[4]:= binconstraints = Map[#^2 == # &, vars]
```

```
Out[4]= {r(1)^2 == r(1), r(2)^2 == r(2), r(3)^2 == r(3), r(4)^2 == r(4), g(1)^2 == g(1),
         g(2)^2 == g(2), g(3)^2 == g(3), g(4)^2 == g(4), s(1)^2 == s(1), s(2)^2 == s(2),
         s(3)^2 == s(3), s(4)^2 == s(4), t(1)^2 == t(1), t(2)^2 == t(2), t(3)^2 == t(3),
         t(4)^2 == t(4), p(1)^2 == p(1), p(2)^2 == p(2), p(3)^2 == p(3), p(4)^2 == p(4)}
```

Set up the equations by translating each of the requirements.

```
In[5]:= problem =
        With[{Adele = 1, Betty = 2, Carol = 3, Doris = 4}, Join[binconstraints, {
            greeneye.slender == 3,
            redhair.tall == 2,
            slender.tall == 2,
            greeneye.redhair == 1,
            greeneye[[Adele]] == greeneye[[Betty]],
            redhair[[Betty]] == redhair[[Carol]],
            slender[[Carol]] == 1 - slender[[Doris]],
            tall[[Adele]] == tall[[Doris]],
            Thread[redhair greeneye slender tall == perfectgirl]]];
```

We solve the system of equations using **NSolve** (this is *much* faster than using **Solve**), obtaining two solutions.

```
In[6]:= sol = NSolve[problem, vars] // Union
```

```
Out[6]= {{r(1) -> 1., r(2) -> 0., r(3) -> 0., r(4) -> 1., g(1) -> 1., g(2) -> 1.,
         g(3) -> 1., g(4) -> 0., s(1) -> 1., s(2) -> 1., s(3) -> 1., s(4) -> 0., t(1) -> 1.,
         t(2) -> 0., t(3) -> 1., t(4) -> 1., p(1) -> 1., p(2) -> 0., p(3) -> 0., p(4) -> 0.},
        {r(1) -> 1., r(2) -> 0., r(3) -> 0., r(4) -> 1., g(1) -> 1., g(2) -> 1., g(3) -> 1.,
         g(4) -> 0., s(1) -> 1., s(2) -> 1., s(3) -> 1., s(4) -> 0., t(1) -> 1., t(2) -> 1.,
         t(3) -> 0., t(4) -> 1., p(1) -> 1., p(2) -> 0., p(3) -> 0., p(4) -> 0.}}
```

However, there is only one perfect girl.

```
In[7]:= Cases[sol, (p(a_) -> 1.) -> women[a], ∞] // Union
Out[7]= {Adele}
```

There are two solutions because the heights of **Betty** and **Carol** are not determined.

```
In[8]:= Select[Union /@ Transpose[sol], Length[#] > 1 &]
Out[8]= {{t(2) -> 0., t(2) -> 1.}, {t(3) -> 0., t(3) -> 1.}}
In[9]:= % /. t(a_) -> women[a]
Out[9]= {{Betty -> 0., Betty -> 1.}, {Carol -> 0., Carol -> 1.}}
```

■ Expectation Value of the Radioactive Decay Constant

Andreas Dieckmann

dieckman@physik.uni-bonn.de

In radioactive decay the probability of the decay of a single particle after a time t is given by an exponential distribution $p(t) = e^{-t/\tau} / \tau$ with τ being the *mean lifetime* characteristic of the kind of particle under consideration. Alternatively, the probability may be expressed by using the *decay constant* $\lambda = 1/\tau$: $p(t) = \lambda e^{-\lambda t}$. Imagine that n particles decay at times t_i , $i = 1, \dots, n$. The probability of a given set of times t_i can be determined from the likelihood function,

$$L = \prod_{i=1}^n \frac{e^{-t_i/\tau}}{\tau} = \frac{e^{-(\sum_{i=1}^n t_i)/\tau}}{\tau^n}, \quad (1)$$

or, in *Mathematica* notation,

$$\text{In}[1]:= L = \frac{e^{-(\sum_{i=1}^n t_i)/\tau}}{\tau^n};$$

The “best” estimator $\hat{\tau}$ (calculated from the measured times) of the true τ is then found according to maximum likelihood (ML) theory: equate the derivative $\partial_{\tau} \log(L)$ at $\tau = \hat{\tau}$ to zero.

$$\begin{aligned} \text{In}[2]:= & \frac{\partial \log(L)}{\partial \tau} == 0 /. \tau \rightarrow \hat{\tau} // \text{ExpandAll} \\ \text{Out}[2]= & \frac{\sum_{i=1}^n t_i}{\hat{\tau}^2} - \frac{n}{\hat{\tau}} == 0 \end{aligned}$$

Then solve for $\hat{\tau}$.

$$\begin{aligned} \text{In}[3]:= & \text{Solve}[\%, \hat{\tau}] \\ \text{Out}[3]= & \left\{ \left\{ \hat{\tau} \rightarrow \frac{\sum_{i=1}^n t_i}{n} \right\} \right\} \end{aligned}$$

The result is that $\hat{\tau}$ is the *arithmetic mean* of the t_i .

To get the *expectation value* for this statistic, $\langle \hat{\tau} \rangle$, compute the integral

$$\langle \hat{\tau} \rangle = \int_0^\infty \dots \int_0^\infty \hat{\tau} L dt_1 \dots dt_n = \frac{1}{n \tau^n} \int_0^\infty \dots \int_0^\infty \left(\sum_{i=1}^n t_i \right) e^{-(\sum_{i=1}^n t_i)/\tau} dt_1 \dots dt_n. \quad (2)$$

Since

$$\text{In}[4]:= \text{Assuming}[\tau > 0, \int_0^\infty e^{-t_i/\tau} dt_i]$$

$$\text{Out}[4]= \tau$$

we have that

$$\int_0^\infty \dots \int_0^\infty e^{-(\sum_{i=1}^n t_i)/\tau} dt_1 \dots dt_n = \prod_{i=1}^n \underbrace{\int_0^\infty e^{-t_i/\tau} dt_i}_{\tau} = \tau^n. \quad (3)$$

The factor $\sum_{i=1}^n t_i$ in equation (2) can be obtained by *parametric differentiation* of equation (3) with respect to τ .

$$\text{In}[5]:= \tau^2 \partial_\tau e^{-(\sum_{i=1}^n t_i)/\tau} // \text{Simplify}$$

$$\text{Out}[5]= e^{-\frac{\sum_{i=1}^n t_i}{\tau}} \sum_{i=1}^n t_i$$

Hence

$$\begin{aligned} \tau^2 \partial_\tau \underbrace{\int_0^\infty \dots \int_0^\infty e^{-(\sum_{i=1}^n t_i)/\tau} dt_1 \dots dt_n}_{\tau^n} &= \underbrace{\int_0^\infty \dots \int_0^\infty \left(\sum_{i=1}^n t_i \right) e^{-(\sum_{i=1}^n t_i)/\tau} dt_1 \dots dt_n}_{n \tau^n \langle \hat{\tau} \rangle} \\ \Rightarrow \langle \hat{\tau} \rangle &= \frac{1}{n \tau^n} \tau^2 \partial_\tau \tau^n = \tau. \end{aligned} \quad (4)$$

So the prescription of calculating the value of τ from the measured times is a good one which is, of course, well known and found in many textbooks [2].

Similarly, it is easy to show that the ML estimator of the decay constant, $\hat{\lambda}$, is simply $\hat{\lambda} = 1 / \hat{\tau}$.

$$\text{In}[6]:= L = \text{Simplify}[L /. \tau \rightarrow \frac{1}{\lambda}, \lambda > 0]$$

$$\text{Out}[6]= e^{-\lambda \sum_{i=1}^n t_i} \lambda^n$$

$$\text{In}[7]:= \frac{\partial \log(L)}{\partial \lambda} == 0 /. \lambda \rightarrow \hat{\lambda} // \text{ExpandAll}$$

$$\text{Out}[7]= \frac{n}{\hat{\lambda}} - \sum_{i=1}^n t_i == 0$$

In[8]:= **Solve**[% , $\hat{\lambda}$]

Out[8]= $\{\{\hat{\lambda} \rightarrow \frac{n}{\sum_{i=1}^n t_i}\}\}$

To compute the expectation value of $\hat{\lambda}$,

$$\langle \hat{\lambda} \rangle = \int_0^\infty \dots \int_0^\infty \hat{\lambda} L dt_1 \dots dt_n = n \lambda^n \int_0^\infty \dots \int_0^\infty \frac{e^{-\lambda \sum_{i=1}^n t_i}}{\sum_{i=1}^n t_i} dt_1 \dots dt_n, \quad (5)$$

we again use *parametric differentiation*,

$$-\partial_\lambda \left(\frac{\langle \hat{\lambda} \rangle}{n \lambda^n} \right) = \int_0^\infty \dots \int_0^\infty e^{-\lambda \sum_{i=1}^n t_i} dt_1 \dots dt_n = \prod_{i=1}^n \int_0^\infty e^{-\lambda t_i} dt_i. \quad (6)$$

Computing the product of integrals is straightforward.

In[9]:= **Assuming**[$\lambda > 0$, **Simplify**[[$\prod_{i=1}^n \int_0^\infty e^{-\lambda t_i} dt_i$]]]

Out[9]= λ^{-n}

Integrating yields $\langle \hat{\lambda} \rangle$.

In[10]:= $\langle \hat{\lambda} \rangle = -n \lambda^n \int \lambda^{-n} d\lambda // \text{Factor}$

Out[10]= $\frac{n \lambda}{n-1}$

Similarly, to compute the expectation value of $\hat{\lambda}^2$,

$$\langle \hat{\lambda}^2 \rangle = n^2 \lambda^n \int_0^\infty \dots \int_0^\infty \frac{e^{-\lambda \sum_{i=1}^n t_i}}{(\sum_{i=1}^n t_i)^2} dt_1 \dots dt_n, \quad (7)$$

parametric differentiation with respect to λ twice yields

$$\partial_{\{\lambda, 2\}} \left(\frac{\langle \hat{\lambda}^2 \rangle}{n^2 \lambda^n} \right) = \prod_{i=1}^n \int_0^\infty e^{-\lambda t_i} dt_i = \lambda^{-n}. \quad (8)$$

Integrating twice yields $\langle \hat{\lambda}^2 \rangle$.

In[11]:= $\langle \hat{\lambda}^2 \rangle = n^2 \lambda^n \int \int \lambda^{-n} d\lambda d\lambda // \text{Factor}$

Out[11]= $\frac{n^2 \lambda^2}{(n-2)(n-1)}$

Now we can compute the variance of $\hat{\lambda}$,

$$\sigma^2(\hat{\lambda}) = \langle (\hat{\lambda} - \langle \hat{\lambda} \rangle)^2 \rangle = \langle \hat{\lambda}^2 \rangle - \langle \hat{\lambda} \rangle^2. \quad (9)$$

In[12]:= $\langle \hat{\lambda}^2 \rangle - \langle \hat{\lambda} \rangle^2$ // Simplify

$$\text{Out[12]} = \frac{n^2 \lambda^2}{(n-2)(n-1)^2}$$

The n -dimensional integral in equation (5) represents the expectation value for the estimator of the radioactive decay constant, calculated from n measurements. Scaling the variables in equation (5), $t_i \rightarrow t_i / \lambda$, we see that

$$\begin{aligned} \langle \hat{\lambda} \rangle &= n \lambda^n \int_0^\infty \dots \int_0^\infty \frac{e^{-\lambda \sum_{i=1}^n t_i}}{\sum_{i=1}^n t_i} dt_1 \dots dt_n \xrightarrow{t_i \rightarrow t_i / \lambda} n \\ \lambda \int_0^\infty \dots \int_0^\infty \frac{e^{-\sum_{i=1}^n t_i}}{\sum_{i=1}^n t_i} dt_1 \dots dt_n &= \frac{n \lambda}{n-1}. \end{aligned} \quad (10)$$

From this we obtain the following identity, which is independent of λ ,

$$\frac{n}{n-1} = \int_0^\infty \dots \int_0^\infty \left(\frac{n}{\sum_{i=1}^n t_i} \right) e^{-\sum_{i=1}^n t_i} dt_1 \dots dt_n. \quad (11)$$

To check this result for $n \geq 2$, instead of computing the integral directly via numeric integration, create exponentially distributed random numbers, t_i , and calculate the numerical averages of $n / \sum_{i=1}^n t_i$.

In[13]:= << Statistics`

In[14]:= pts = With[{p = 40000}, Table[{n,

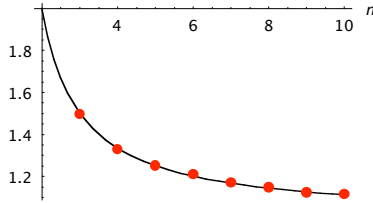
$$\frac{1}{[p/n]} \sum_{j=1}^{p/n} \frac{n}{\sum_{i=1}^n \text{Random}[\text{ExponentialDistribution}[1]]}], \{n, 2, 10\}]]$$

$$\text{Out[14]} = \begin{pmatrix} 2 & 2.08508 \\ 3 & 1.4963 \\ 4 & 1.3286 \\ 5 & 1.25124 \\ 6 & 1.20953 \\ 7 & 1.17065 \\ 8 & 1.148 \\ 9 & 1.12319 \\ 10 & 1.11571 \end{pmatrix}$$

Plotting this data (•) and comparing with the analytical expression $n / (n-1)$, we see that the agreement is excellent.

```
In[15]:= Plot[ $\frac{n}{n-1}$ , {n, 2, 10},
  Epilog -> {Hue[1], AbsolutePointSize[4], Point /@ pts}, AxesLabel -> {n, ""}]
```

From In[15]:=



A general result for a wide class of related integrals involving the k th power of t_i is easily obtained. Since

```
In[16]:= Assuming[ $\lambda > 0 \wedge k > 0$ ,  $\int_0^\infty e^{-\lambda t_i^k} dt_i$ ]
```

```
Out[16]=  $\lambda^{-1/k} \Gamma\left(1 + \frac{1}{k}\right)$ 
```

integrating the product of n such terms yields

$$\int_0^\infty \dots \int_0^\infty e^{-\lambda \sum_{i=1}^n t_i^k} dt_1 \dots dt_n = \left(\lambda^{-1/k} \Gamma\left(1 + \frac{1}{k}\right) \right)^n. \quad (12)$$

Integrating both sides with respect to λ , m times, and using induction on

```
In[17]:=  $\int \lambda^\alpha d\lambda$ 
```

```
Out[17]=  $\frac{\lambda^{\alpha+1}}{\alpha+1}$ 
```

we find the following formula (using $\Gamma(1 + \frac{1}{k}) = \frac{1}{k} \Gamma(\frac{1}{k})$),

$$\left(\frac{k \lambda^{1/k}}{\Gamma(\frac{1}{k})} \right)^n \int_0^\infty \dots \int_0^\infty \frac{e^{-\lambda \sum_{i=1}^n t_i^k}}{(\sum_{i=1}^n t_i^k)^m} dt_1 \dots dt_n = \lambda^m \frac{\Gamma(\frac{n}{k} - m)}{\Gamma(\frac{n}{k})}, \quad (13)$$

valid for $\lambda > 0$, $n > k m > 0$, $k > 0$. This result holds for nonintegral m and under weaker conditions on λ , n , k , and m . For example, with $n = 1$,

```
In[18]:= Assuming[ $\text{Re}(\lambda) > 0 \wedge \text{Re}(k) > 0 \wedge \text{Re}(k m) < 1$ ,
```

$$\left(\frac{k \lambda^{1/k}}{\Gamma(\frac{1}{k})} \right) \int_0^\infty \frac{e^{-\lambda t^k}}{t^{k m}} dt = \lambda^m \frac{\Gamma(\frac{1}{k} - m)}{\Gamma(\frac{1}{k})}]$$

```
Out[18]= True
```

Scaling the variables in equation (13), $t_i \rightarrow t_i / \lambda$, is equivalent to putting $\lambda \rightarrow 1$, that is

$$\left(\frac{k}{\Gamma(\frac{1}{k})} \right)^n \int_0^\infty \dots \int_0^\infty \left(\sum_{i=1}^n t_i^k \right)^{-m} e^{-\sum_{i=1}^n t_i^k} dt_1 \dots dt_n = \frac{\Gamma(\frac{n}{k} - m)}{\Gamma(\frac{n}{k})}. \quad (14)$$

Noting that

$$\text{In}[19]:= \text{FullSimplify}\left[\left(\frac{k}{\Gamma(\frac{1}{k})}\right) \int_0^\infty e^{-t^k} dt = Q\left(\frac{1}{k}, 0, z^k\right), z > 0 \wedge k > 0\right]$$

`Out[19]= True`

where Q is the generalized regularized incomplete gamma function, we can test equation (14) for $k \geq 2$ with random numbers t_i^k distributed according to the inverse of Q , and calculating the numerical averages of $(\sum_{i=1}^n t_i^k)^{-m}$ for a range of $n > km$.

Generating inverse values of this distribution is time consuming, so we save the computed values using dynamic programming.

`In[20]:= rs[k_] := rs[k] = Table[Q-1(1/k, 0, Random[]), {40000}];`

Use **Partition**, **Tr** (twice), and **Length** to compute the averages efficiently. Here are the results for $k = 3$ and $m = 2$.

`In[21]:= pts = With[{k = 3, m = 2},`

$$\text{Table}\left[\left\{n, \text{With}\left[\{p = \text{Partition}[\text{rs}[k], n]\}, \frac{1}{\text{Length}[p]} \text{Tr}[(\text{Tr} /@ p)^{-m}]\right\}, \right.$$

$$\left. \{n, km + 1, km + 10\}\right]$$

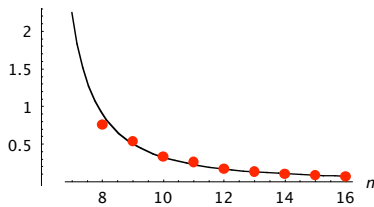
`Out[21]=`

7	2.54078
8	0.756443
9	0.5366
10	0.33347
11	0.257472
12	0.16879
13	0.129687
14	0.100626
15	0.085662
16	0.0690711

Plotting this data (•) and comparing with the analytical expression $\Gamma(n/k - m) / \Gamma(n/k)$, again we see that the agreement is excellent.

```
In[22]:= With[{k = 3, m = 2}, Plot[ $\Gamma(n/k - m) / \Gamma(n/k)$ , {n, k m + 1, k m + 10},
  Epilog -> {Hue[1], AbsolutePointSize[4], Point /@ pts},
  AxesOrigin -> {k m, 0}, AxesLabel -> {n, ""}]]
```

From In[22]:=



■ References

- [1] H. J. Jeffrey, "Chaos Game Representation of Gene Structure," *Nucleic Acids Research*, **18**, 1990 pp. 2163–2175.
- [2] A. G. Frodeson, O. Skjeggstad, and H. Tofte, *Probability and Statistics in Particle Physics*, New York: Columbia University, 1979.

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

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* news group, comp.soft-sys.math.mathematica, that appear here are edited for clarity and length.

■ StepMonitor

Q: I am running a complex **FindMinimum** calculation that takes a long time to run. If I abort the computation, is there a way to output the last fully solved iteration that **FindMinimum** has found?

A: Robert Knapp (rknapp@wolfram.com) answers: In Version 5, you can use the **StepMonitor** option to keep track of the iterations.

Here is a simple function that uses **Pause** to simulate running a complex computation.

```
In[1]:= f[x_?NumberQ, y_] := (Pause[1]; (x - 1)^2 + (x^2 - y)^2)
```

StylePrint is used to provide intermediate results as **Output** style cells.

```
In[2]:= FindMinimum[f[x, y], {{x, -2}, {y, 1}},  
  MaxIterations -> 3, StepMonitor -> StylePrint[{x, y}, "Output"]  
  
{-0.903677, 1.21926}  
  
{-0.808228, 1.18443}  
  
{0.475604, 0.501006}
```

```
FindMinimum::cvmit :  
  Failed to converge to the requested accuracy or precision within 3 iterations. More...
```

```
Out[2]= {0.35051, {x -> 0.475604, y -> 0.501006}}
```

To get a result when you abort the computation, you need to use **CheckAbort**.

```
In[3]:= CheckAbort[FindMinimum[f[x, y],
    {{x, -2}, {y, 1}}, StepMonitor :> (laststep = {x, y})], laststep]
```

```
Out[3]= {2.54791 × 10-15, {x → 1., y → 1.}}
```

If the problem runs to completion, you will get the result from **FindMinimum**. If you abort the computation, you will obtain the last step taken in the iteration.

When **FindMinimum** gets close to a solution, differences between iterations do not show up in **Output** style cells. One can output results with more digits by changing the **PrintPrecision** display option as follows:

1. Open the Option Inspector by clicking Format ▸ Option Inspector. Set the scope to notebook (or global).
2. Locate **PrintPrecision** by typing “printp” in the Lookup text field.
3. Set **PrintPrecision** to the number of digits you want displayed.

■ Beeping Progress

Q: How can I play a sound to alert me regarding my program’s progress during a long looping procedure?

A: You can use the built-in **Message** mechanism. You can specify the required (default) behavior of **Message** by setting **WarningAction** to include **Beep** as follows:

1. Open the Option Inspector by clicking Format ▸ Option Inspector. Make sure the scope is global.
2. Locate **WarningAction** by typing in “warning” in the Lookup text field.
3. Modify **WarningAction** to include **Beep**.

Add a message displaying the iteration number (see Section 2.9.21 of *The Mathematica Book*).

```
In[1]:= Progress::update = "Iteration `1`.";
```

Redundant or repetitive messages are, by default, suppressed in long calculations; therefore, you should turn off **General::stop** (see **General::stop** in the Reference Guide).

```
In[2]:= Off[General::stop]
```

From your loop, call this message.

```
In[3]:= Table[If[Mod[i, 200] == 0, Message[Progress::update, i]]; {i, ζ(i)}, {i, 1000}];
```

Progress::update : Iteration 200.

Progress::update : Iteration 400.

Progress::update : Iteration 600.

Progress::update : Iteration 800.

Progress::update : Iteration 1000.

Alternatively, if you have suitable text-to-speech system software installed on your system, you can ask the front end to “speak” the given text, which should be a text string.

```
In[4]:= Table[If[Mod[i, 100] == 0,
  FrontEndExecute[SpeakTextPacket["Iteration " <> ToString[i]]];
  {i, ζ(i)}, {i, 1100, 1500}];
```

To set the speaking voice, use **FrontEndExecute[SetSpeechParameters`
Packet[i]]** where *i* is a number between 1 and the number of voices you have installed. (For further information, see the following *The Mathematica Journal* articles: “Tricks of the Trade” (SetNotebookStatusLine), 7 (3), 1999 p. 263, and “In and Out” (Why the Beep?), 8 (3), 2002 p. 400.)

■ Coplanarity

Q: Given a set of *n*-dimensional vectors, is there a simple way to check if they are coplanar in $\mathbb{R}^{n-1}$?

A: Eckhard Hennig (aidev@kaninkolo.de) answers: To establish coplanarity in $\mathbb{R}^{n-1}$ of a set of *n*-dimensional vectors, *V*, you need to check whether the dimension of the null space (or *nullity*) of the linear mapping defined by *V* has dimension ≥ 1 . Here is an example.

Define three linearly independent vectors in $\mathbb{R}^3$.

```
In[1]:= a1 = {2, 2, 3}; a2 = {1, -1, 2}; a3 = {1, 0, -1};
```

Generate a set of 10 coplanar vectors in $\mathbb{R}^3$ by forming random linear combinations of *a*₁ and *a*₂. Use **SeedRandom** to obtain a repeatable sequence of pseudo-random numbers.

```
In[2]:= SeedRandom[1234];
```

```
In[3]:= v2 = Table[Table[Random[Real, {-10, 10}], {2}].{a1, a2}, {10}]
```

```
Out[3]=
```

$$\begin{pmatrix} 14.3308 & 0.162139 & 25.0384 \\ 5.2323 & -10.6653 & 11.8228 \\ 15.3128 & 15.8908 & 22.8247 \\ -6.12104 & 10.531 & -13.3446 \\ -23.3773 & -15.7515 & -36.9723 \\ -0.314199 & 10.456 & -3.16384 \\ 2.51425 & -8.06091 & 6.41517 \\ 11.8982 & 12.8355 & 17.6129 \\ 6.58317 & 4.87541 & 10.3017 \\ -3.20984 & 8.46853 & -7.73436 \end{pmatrix}$$

The null space of the mapping defined by v_2 has dimension 1; hence, the set is coplanar.

```
In[4]:= Length[NullSpace[v2]]
```

```
Out[4]= 1
```

In Version 5, you can use **MatrixRank** instead to calculate the dimension of the subspace spanned by v_2 . The vectors are coplanar if the rank of v_2 is 2 (or $n - 1$ in the general case).

```
In[5]:= MatrixRank[v2]
```

```
Out[5]= 2
```

NullSpace may be unreliable for large and ill-conditioned systems. A more reliable, but less efficient, alternative is to determine the number of singular values of the mapping.

```
In[6]:= Length[SingularValueList[v2]]
```

```
Out[6]= 2
```

Since there are two singular values, the set of vectors spans a two-dimensional space, that is, the vectors are coplanar in $\mathbb{R}^3$.

Now crosscheck for the noncoplanar case: generate 10 vectors in $\mathbb{R}^3$ by forming random linear combinations of the three independent basis vectors a_1 , a_2 , and a_3 .

```
In[7]:= v3 = Table[Table[Random[Real, {-10, 10}], {3}].{a1, a2, a3}, {10}]
```

```
Out[7]= {
  11.7238    10.4453    15.8459
  13.4381    26.7939     8.07186
 -30.5407   -7.09987   -34.9813
  -2.41632  -14.0657   -27.2224
   3.17777  -19.8561   -12.7652
   7.14493   4.95061   33.2804
 -27.6994  -10.9223   -24.3901
  -5.41409   1.31064   14.0318
 -25.1416    2.88407  -17.8299
 -23.0158  -11.0079   -31.1886
}
```

The null space is empty, which implies that a_1 , a_2 , and a_3 span $\mathbb{R}^3$.

```
In[8]:= NullSpace[v3]
```

```
Out[8]= {}
```

Alternatively, the number of singular values equals the dimension of the space.

```
In[9]:= Length[SingularValueList[v3]] == 3
```

```
Out[9]= True
```

■ Autocompilation

Q: I would like to understand the timing difference between the following two code samples. Slow version:

```
In[1]:= n = m = 100; First@Timing[Do[Table[1, {i, 1, n}, {j, 1, m}], {100}];]
```

```
Out[1]= 2.41 Second
```

Fast version:

```
In[2]:= First@Timing[Do[Table[1, {i, 1, 100}, {j, 1, 100}], {100}];]
```

```
Out[2]= 0.46 Second
```

Is this due to autocompilation?

A: Robert Knapp (rknapp@wolfram.com) answers: The issue has to do with autocompilation, but is quite subtle. `Table[1, {i, 1, n}, {j, 1, m}]` is basically equivalent to `Table[Table[1, {j, 1, m}], {i, 1, n}]`. With the second form, it is a bit easier to see why it cannot be autocompiled. Each time the inner **Table** is computed, *m* needs to be evaluated to determine what the limit should be. Therefore the issue boils down to why it needs to be evaluated *each* time through

the outer loop. Before evaluation, m is just a symbol, so it could have any evaluation attached to it, for example, $m := 10 i$. In this example, the result of evaluating with only one i would obviously be wrong. You might think it would be possible to evaluate with i unset, but there are cases where that would be incorrect, for example, $m := \text{If}[\text{NumberQ}[i], 10 i, 0]$. This is, of course, pathological, but it is not too difficult to construct real cases that give incorrect or misleading values. It is very important that *Mathematica* be consistent, so we cannot do this. Thus, since m cannot be known, except for specific values of i , there is no way to autocompile the loop as a whole in a really efficient way. On the other hand, when you evaluate or put in literal numbers, everything is known, so compilation analysis can be done quite completely.

■ Sparse Matrices

Q: One can use **LinearSolve** to solve a system of linear equations. However, I work with very large matrices having only five nonzero diagonals. Are there efficient methods for dealing with such matrices?

A: Daniel Lichtblau (danl@wolfram.com) answers: If you require only machine arithmetic, this can be done effectively using the built-in sparse linear solver in Version 5. Here is an example.

Create a diagonally-dominant random sparse $n \times n$ d -diagonal matrix using **SparseArray**.

```
In[1]:= SeedRandom[1111];
In[2]:= RandomMatrix[n_, d_?OddQ] :=
  SparseArray[{{i_, j_} /; |i - j| ≤ (d - 1) / 2 → δi,j + Random[], {n, n}]
```

With $n = 4$ and $d = 3$, we obtain a sparse array with 10 nonzero elements.

```
In[3]:= RandomMatrix[4, 3]
Out[3]= SparseArray[<10>, {4, 4}]
```

Use **Normal** to obtain the ordinary array corresponding to a sparse array object.

```
In[4]:= % // Normal
Out[4]= 
$$\begin{pmatrix} 1.93339 & 0.814818 & 0 & 0 \\ 0.688359 & 1.1433 & 0.833 & 0 \\ 0 & 0.317132 & 1.91055 & 0.346584 \\ 0 & 0 & 0.0287865 & 1.64717 \end{pmatrix}$$

```

Produce a pentadiagonal sparse matrix of dimension 2000.

```

In[5]:= m = 2000; mat = RandomMatrix[m, 5]
Out[5]= SparseArray[<9994>, {2000, 2000}]

Solve for a random right-hand side.

In[6]:= rhs = Table[Random[], {m}];
In[7]:= Timing[sol1 = LinearSolve[SparseArray[mat], rhs];] // First
Out[7]= 0.09 Second

Compare with the time required for an ordinary (nonsparse) matrix.

In[8]:= Timing[sol2 = LinearSolve[Normal[SparseArray[mat]], rhs];] // First
Out[8]= 18.95 Second

Check that the solutions agree using Norm (for matrices, Norm[m] gives the
maximum singular value of m).

In[9]:= Norm[sol2 - sol1]
Out[9]= 4.29374 × 10-11

```

■ ϵ - δ Proofs

Q: How can I do ϵ - δ proofs for limits? For example, prove that the limit of $\frac{x^2-4}{x-2}$, as x approaches 2, is 4.

A: Andrzej Kozłowski (akoz@mimuw.edu.pl) answers: If you restrict attention to rational functions, you can use quantifiers and **Resolve** to do this.

```

In[1]:= Resolve[ForAll[epsilon>0, (Exists[delta>0, (ForAll[x, Abs[x-2]<delta& x in R& lambda in R, Abs[frac{x^2-4}{x-2} - lambda] < epsilon)])]]
Out[1]= lambda == 4

```

■ Integer Solutions

Q: Is there a simple way to compute the solutions to $p + q = n$ for $2 \leq p \leq \frac{n}{2}$, $p \leq q$, $p, q \in \mathbb{P}$ for a specified $n > 4$, $n \in \mathbb{Z}$?

A: Adam Strzebonski (adams@wolfram.com) answers: In Version 5, you can use **Reduce**, though it requires changing the value of a system option. By default,

Reduce writes out integer solutions of $a \leq x \leq b$ explicitly if the number of integers in $[a, b]$ is less than 11.

In[1]:= **Reduce**[$1 \leq x \leq 10, x, \mathbb{Z}$]

Out[1]= $x = 1 \vee x = 2 \vee x = 3 \vee x = 4 \vee x = 5 \vee x = 6 \vee x = 7 \vee x = 8 \vee x = 9 \vee x = 10$

Otherwise, the solutions are represented as $x \in \mathbb{Z} \wedge a \leq x \leq b$.

In[2]:= **Reduce**[$1 \leq x \leq 11, x, \mathbb{Z}$]

Out[2]= $x \in \mathbb{Z} \wedge 1 \leq x \leq 11$

Here is the general *parametric* solution to the given problem with $n = 500$.

In[3]:= **m1**[n_] := **Reduce**[$p + q = n \wedge 2 \leq p \leq \frac{n}{2} \wedge p \leq q, \{p, q\}, \mathbb{P}$]

In[4]:= **m1**[500]

Out[4]= $(p \mid q) \in \mathbb{P} \wedge c_1 \in \mathbb{Z} \wedge 2 \leq c_1 \leq 250 \wedge p = c_1 \wedge q = 500 - c_1$

Changing the value of a **Reduce** system option yields *explicit* solutions.

In[5]:= **Developer`SetSystemOptions**[
"ReduceOptions" $\rightarrow$ "DiscreteSolutionBound" $\rightarrow \infty$];

In[6]:= **Reduce**[$1 \leq x \leq 11, x, \mathbb{Z}$]

Out[6]= $x = 1 \vee x = 2 \vee x = 3 \vee x = 4 \vee x = 5 \vee$
 $x = 6 \vee x = 7 \vee x = 8 \vee x = 9 \vee x = 10 \vee x = 11$

Here are all the solutions to the given problem with $n = 500$.

In[7]:= **m1**[500]

Out[7]= $p = 13 \wedge q = 487 \vee p = 37 \wedge q = 463 \vee p = 43 \wedge q = 457 \vee p = 61 \wedge q = 439 \vee$
 $p = 67 \wedge q = 433 \vee p = 79 \wedge q = 421 \vee p = 103 \wedge q = 397 \vee$
 $p = 127 \wedge q = 373 \vee p = 151 \wedge q = 349 \vee p = 163 \wedge q = 337 \vee$
 $p = 193 \wedge q = 307 \vee p = 223 \wedge q = 277 \vee p = 229 \wedge q = 271$

Although this approach is simplest in that it requires just a single command and no manual preprocessing of the input, it is not very efficient. **Reduce** has no special methods for solving equations and inequalities over the primes, other than by generating primes in a given finite interval. If we give the problem to **Reduce** directly, it uses a general method for solving systems of equations and inequalities over the integers, and then removes the numeric solutions that are not prime. Changing the system options causes $2 \leq c_1 \leq 250$ to be represented as 249 explicit values, from which we get 249 integer solutions for p and q , and only then does it remove those that are not prime.

A more efficient way of solving this particular problem is to make **Reduce** generate only prime values for p , and then select those for which $q = n - p$ is prime.

In[8]:= **m2**[n_] := **Or** @@ ($p = \# \wedge q = n - \# \& /@$

Select[$p /. \{\text{ToRules}[\text{Reduce}[2 \leq p \leq \frac{n}{2}, p, \mathbb{P}]]\}, \text{PrimeQ}[n - \#] \&]$)

Here is a comparison of the two methods.

```
In[9]:= (s1 = m1[105];) // Timing // First
Out[9]= 124.31 Second

In[10]:= (s2 = m2[105];) // Timing // First
Out[10]= 0.35 Second

In[11]:= s1 === s2
Out[11]= True
```

■ Interpolation

Q : Given a set of nonuniformly sampled $\{x, y\}$ data, how can I compute numerical values of the slope dy/dx , second derivative d^2y/dx^2 , and integral $\int y^2 dx$? Also, is it possible to compute y as a function of uniformly incremented arc length?

A : Here is an exact function $y(x) = x \sin(x) - x/2$.

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

Sample $y(x)$, taking x from a uniform random distribution over the interval $(a, b) = (1, 12)$, and append the function values at the endpoints.

```
In[2]:= {a, b} = {1, 12};
```

```
In[3]:= f(x_) = {x, y(x)};
```

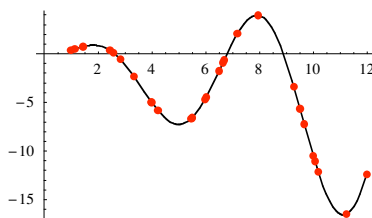
```
In[4]:= SeedRandom[4321];
```

```
In[5]:= data = N[Sort[f /@ Join[{a}, Table[Random[Real, {a, b}], {25}], {b}]]];
```

Superimpose the sampled data over a plot of the function to visualize the $\{x, y\}$ data.

```
In[6]:= Plot[y(x), {x, a, b}, Epilog -> {Hue[1], AbsolutePointSize[3], Point /@ data}]
```

From In[6]:=



Interpolate the data using **Interpolation** of sufficiently high order.

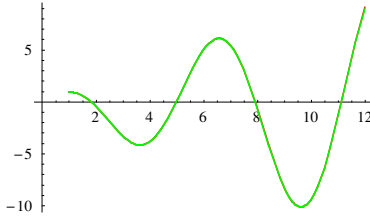
```
In[7]:= y_int = Interpolation[data, InterpolationOrder → 5]
```

```
Out[7]= InterpolatingFunction[{{1. 12.}}, <>]
```

Compare the first and second derivatives; the red curve denotes the exact value and the green curve denotes the interpolated function in each plot. The agreement is very good.

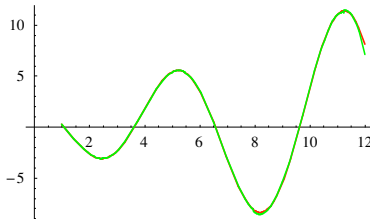
```
In[8]:= Plot[{y'(x), y'_int(x)}, {x, a, b}, PlotStyle → {Hue[0], Hue[ $\frac{1}{3}$ ]}
```

From In[8]:=



```
In[9]:= Plot[{y''(x), y''_int(x)}, {x, a, b}, PlotStyle → {Hue[0], Hue[ $\frac{1}{3}$ ]}
```

From In[9]:=



Next, examine $\int_a^x y(t)^2 dt$. For comparison purposes, compute the exact value of $\int_a^x y(t)^2 dt$.

```
In[10]:= i_exact(x_) =  $\int_a^x y(t)^2 dt$ 
```

```
Out[10]=  $\frac{x^3}{4} + \cos(x) x^2 - \frac{1}{4} \sin(2x) x^2 - \frac{1}{4} \cos(2x) x - 2 \sin(x) x -$   

 $2 \cos(x) + \frac{1}{8} \sin(2x) + \frac{\sin(2)}{8} + 2 \sin(1) + \frac{\cos(2)}{4} + \cos(1) - \frac{1}{4}$ 
```

Compute the numerical value of $\int_a^x y(t)^2 dt$ as an **InterpolatingFunction** using **FunctionInterpolation** and **Integrate**.

```
In[11]:= i_int(x_) =  $\int$  FunctionInterpolation[y_int(x)^2, {x, a, b}][x] dx
```

```
Out[11]= InterpolatingFunction[{{1. 12.}}, <>][x]
```

Alternatively, compute the indefinite integral as an **InterpolatingFunction** using **NDSolve** (since $g(x) = \int_a^x y(t)^2 dt$ is equivalent to $g'(x) = y(x)^2$ with $g(a) = 0$).

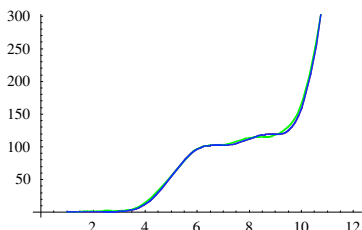
```
In[12]:= i_ndsolv(x_) = g(x) /. First[NDSolve[{y_int'(x) == g'(x), g(a) == 0}, g, {x, a, b}]]
```

```
Out[12]= InterpolatingFunction[({ 1. 12. }, <>)][x]
```

Compare the exact value i_{exact} (red) to the two numerical approaches, i_{int} (green) and i_{ndsolv} (blue). The curves overlay one another quite well.

```
In[13]:= Plot[{i_exact(x), i_int(x), i_ndsolv(x)}, {x, a, b},
  PlotStyle -> {Hue[0], Hue[1/3], Hue[2/3]}]
```

From In[13]:=



The arc length, $r(x)$, of a parametric curve is

$$r(x) = \int_a^x \sqrt{1 + y'(t)^2} dt,$$

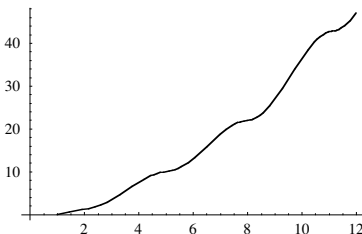
so we have $r'(x) = \sqrt{1 + y'(x)^2}$ with $r(a) = 0$. Here is the arc length, r , as an **InterpolatingFunction**.

```
In[14]:= r = r /. First[NDSolve[{r'(x) == Sqrt[y_int'(x)^2 + 1], r(a) == 0}, r, {x, a, b}]];
```

The arc length is, as expected, monotonically increasing.

```
In[15]:= Plot[r(x), {x, a, b}]
```

From In[15]:=



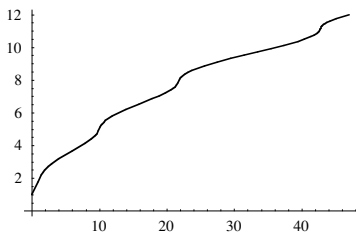
Since $x'(r) = 1/r'(x)$ we can also determine the inverse function $x(r)$.

```
In[16]:= x = x /. First[NDSolve[{1/x'(r) == Sqrt[1 + y_int'(x(r))^2], x(0) == a}, x, {r, r(a), r(b)}]];
```

Here is a plot of $x(r)$.

In[17]:= **Plot**[$x(r)$, { r , $r(a)$, $r(b)$ }]

From In[17]:=



Here is a table of uniformly incremented arc length values of the interpolated function y_{int} .

In[18]:= **arcdata** = **Table**[{ $x(t)$, $y_{\text{int}}(x(t))$ }, **Evaluate**[{ t , $r(a)$, $r(b)$, $\frac{r(b) - r(a)}{20}$ }]]

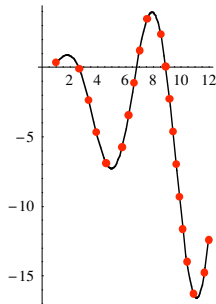
Out[18]=

1.	0.341471
2.67154	-0.125759
3.35219	-2.37684
3.90786	-4.66386
4.61717	-6.9037
5.76061	-5.75552
6.2294	-3.44957
6.61897	-1.12848
7.01705	1.19114
7.56797	3.47666
8.53554	2.37315
8.89554	0.0476488
9.17178	-2.28962
9.41651	-4.63042
9.64925	-6.97245
9.88311	-9.31438
10.1326	-11.6547
10.4256	-13.9899
10.8973	-16.292
11.6967	-14.7726
12.	-12.4389

A **Plot** with **AspectRatio** $\rightarrow$ **Automatic** shows that the sampling does appear to be uniform with respect to arc length.

```
In[19]:= Plot[yint(x), {x, a, b},
  Epilog -> {Hue[1], AbsolutePointSize[3], Point /@ arcdata},
  AspectRatio -> Automatic]
```

From In[19]:=



■ NonlinearRegress

Q : Consider the following initial-value problem.

```
In[1]:= de[a_, b_][t_] = y'(t) == a - b y(t)^3;
```

```
In[2]:= ic = y(0) == 1;
```

The differential equation cannot be solved explicitly so you have to use **NDSolve**. How can I use the output of **NDSolve** as the model function argument to **NonlinearRegress**?

A : Carl Woll (carlw@u.washington.edu) writes: Basically, there are four steps.

1. Define the model with both the parameters and variables restricted to numeric quantities to prevent **NDSolve** from being given symbolic input.
2. Compute the derivatives of the model with respect to each parameter.
3. Teach *Mathematica* about these derivatives.
4. Use **NonlinearRegress** as usual.

After restricting the input parameters a and b to be numeric quantities, numerical solution is straightforward. The variable t used inside **NDSolve** is in a **Module** to ensure that it does not already have a value. Dynamic programming improves efficiency by computing and saving the solution for fixed a and b .

```

In[3]:= f[a_?NumericQ, b_?NumericQ] :=
  f[a, b] = Module[{t}, y /. First[NDSolve[{de[a, b][t], ic}, y, {t, 0, 1}]]]

```

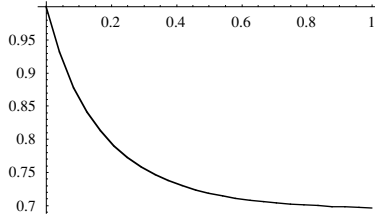
Plot the solution for $a = 1$ and $b = 2$.

```

In[4]:= Plot[f[1, 2][t], {t, 0, 1}]

```

From In[4]:=



Next, compute the derivatives of $f[a, b][t]$, with respect to the *parameters* a and b , by differentiating the parameters of the differential equation and the initial condition.

```

In[5]:=  $\frac{db}{da} \wedge 0; \frac{dt}{da} \wedge 1; \text{dea}[a_, b_][t_] = \frac{d \text{de}[a, b][t]}{da} /. y^{(n)}(t_) \rightarrow ya^{(n-1)}(t)$ 

```

```

Out[5]= ya'(t) == 1 - 3 b y(t)^2 ya(t)

```

```

In[6]:=  $\frac{da}{db} \wedge 0; \frac{dt}{db} \wedge 1; \text{deb}[a_, b_][t_] = \frac{d \text{de}[a, b][t]}{db} /. y^{(n)}(t_) \rightarrow yb^{(n-1)}(t)$ 

```

```

Out[6]= yb'(t) == -y(t)^3 - 3 b yb(t) y(t)^2

```

Then use **NDSolve** on the system of ordinary differential equations, consisting of the original and the differentiated equations.

```

In[7]:= fa[a_?NumericQ, b_?NumericQ] :=
  fa[a, b] = Module[{t}, ya /. First[NDSolve[
    {de[a, b][t], dea[a, b][t], ic, ya[0] == 0}, {y, ya}, {t, 0, 1}]]]

```

```

In[8]:= fb[a_?NumericQ, b_?NumericQ] :=
  fb[a, b] = Module[{t}, yb /. First[NDSolve[
    {de[a, b][t], deb[a, b][t], ic, yb[0] == 0}, {y, yb}, {t, 0, 1}]]]

```

After you have computed the derivatives of the model with respect to the parameters, teach *Mathematica* this information.

```

In[9]:= f /:  $\frac{\partial f[a, b][t_]}{\partial a} := fa[a, b][t]$ 

```

```

In[10]:= f /:  $\frac{\partial f[a, b][t_]}{\partial b} := fb[a, b][t]$ 

```

Next, use **NonlinearRegress** in the usual way. Load the package.

```

In[11]:= << Statistics`NonlinearFit`

```

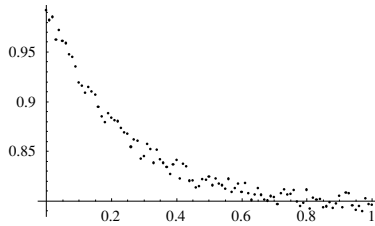
Create some noisy data.

```
In[12]:= SeedRandom[10]
```

```
In[13]:= data = Table[{t, f[1, 2][t] + Random[Real, {-0.01, 0.01}]}, {t, 0, 1, 0.01}];
```

```
In[14]:= lp = ListPlot[data]
```

From In[14]:=



Then run **NonlinearRegress**.

```
In[15]:= NonlinearRegress[data, f[a, b][t], t,  $\begin{pmatrix} a & 0 \\ b & 1 \end{pmatrix}$ ]
```

```
Out[15]= {BestFitParameters → {a → 0.940321, b → 1.89889}, ParameterCITable →
```

	Estimate	Asymptotic SE	CI
a	0.940321	0.0233763	{0.893937, 0.986704},
b	1.89889	0.038682	{1.82213, 1.97564}

```
EstimatedVariance → 0.0000335898, ANOVATable →
```

	DF	SumOfSq	MeanSq
Model	2	71.8804	35.9402
Error	99	0.00332539	0.0000335898,
Uncorrected Total	101	71.8837	
Corrected Total	100	0.277771	

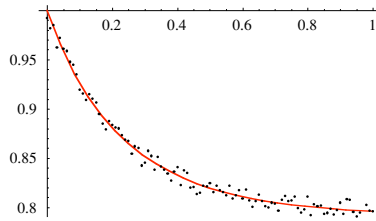
```
AsymptoticCorrelationMatrix →  $\begin{pmatrix} 1. & 0.992543 \\ 0.992543 & 1. \end{pmatrix}$ ,
```

```
FitCurvatureTable →  $\begin{array}{l} \text{Max Intrinsic} \\ \text{Max Parameter-Effects} \\ 95. \% \text{ Confidence Region} \end{array} \begin{array}{l} \text{Curvature} \\ 0 \\ 0 \\ 0.569042 \end{array} \}$ 
```

View the best-fit solution.

```
In[16]:= Plot[f[a, b][t] /. (BestFitParameters /. %),
           {t, 0, 1}, Epilog -> First[lp], PlotStyle -> Hue[1]]
```

From In[16]:=



This method allows you to use the full power of **NonlinearRegress** in a simple and very natural way. There are other approaches, but most of them require that you either use the **FindMinimum** method of **NonlinearRegress**, or that you abandon **NonlinearRegress** (and all of its statistical feedback) and use **FindMinimum** directly.

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*

Modular Equations of the Rogers–Ramanujan Continued Fraction

Michael Trott

In this issue's Corner, we will define and investigate the Rogers–Ramanujan continued fraction. Using high-order series expansions and modular linear algebra allows us to find new two and three-power modular equations for the Rogers–Ramanujan continued fraction. Using the function `Recognize` allows us to determine exact algebraic values for arguments of the form $e^{-\pi\sqrt{r}}$.

■ Introduction

In today's corner, I will do some “experimental” mathematics. This new and exciting field of mathematics uses the unique capabilities of modern mathematical software (see [1, 2, 3] and *Experimental Mathematics* [www.expmath.org] for some examples of experimental mathematics at work).

Starting 100 years ago Rogers [4], Ramanujan, and Schur in Berndt [5] investigated the following continued fraction

$$\mathcal{R}(q) = q^{1/5} \cfrac{1}{1 + \cfrac{q}{1 + \cfrac{q^2}{1 + \cfrac{q^3}{1 + \cfrac{q^4}{1 + \cfrac{q^5}{\dots}}}}}}.$$

For the moment ignoring the factor $q^{1/5}$, this fraction is a natural geometric series analog of the simplest of all possible fractions for the golden ratio

$$\phi = \frac{\sqrt{5} + 1}{2} = \frac{1}{1 + \frac{1}{1 + \frac{1}{1 + \frac{1}{1 + \frac{1}{\dots}}}}}$$

Despite its innocent look, what is now called the Rogers–Ramanujan continued fraction $\mathcal{R}(q)$ still allows for new discoveries. (For various identities, see mathworld.wolfram.com/Rogers-RamanujanContinuedFraction.html.) We will derive some new modular identities for $\mathcal{R}(q)$, mainly using the function `Null`Space` with an explicit setting of the `Modulus` option. We will not prove any of these rigorously (a difficult task), but rather will verify the correctness of the relations to hundreds or thousand of digits using *Mathematica*'s high-precision arithmetic. The probability that thousands of digits of two not specially constructed real numbers agree is astronomically small.

■ Visualizations

```
In[1]:= Off[General::spell]; Off[General::spell1];
$FormatType = TraditionalForm;
$TextStyle = {FontFamily -> "Futura-Book", FontSize -> 6};
$Line = 0;
```

We start with some visualizations to get an impression of the complexity of the Rogers–Ramanujan continued fraction. Unfortunately, because it is not a simple continued fraction, we cannot use built-in functions like `FromContinuedFraction`. So we implement a two-argument version of $\mathcal{R}(q)$ where the second argument indicates the number of divisions.

```
In[1]:= $RecursionLimit = ∞;

In[2]:= R[q_, n_] := q^(1/5) Fold[ $\frac{\#2}{1 + \#1}$  &, q^n, Table[q^k, {k, n - 1, 0, -1}]]
```

Here are the first five levels of the Rogers–Ramanujan continued fraction.

```
In[3]:= R[q, 5]

Out[3]= 
$$\frac{q^{1/5}}{1 + \frac{q}{1 + \frac{q^2}{1 + \frac{q^3}{1 + \frac{q^4}{1 + q^5}}}}}$$

```

Fortunately, for numerical evaluation, the continued fraction form is very convenient (as opposed to the ratio of two polynomials) and stable, and, even for

hundreds of divisions, machine-precision arithmetic is sufficient to get reliable results. (It is important to avoid high-precision arithmetic because we need tens of thousands of values for graphics.) The next input compares a machine- with a high-precision calculation.

```
In[4]:= With[{q =  $\frac{9}{10} e^{2i}$ }, {R[N[q], 500], N[R[N[q], 100], 500]}]
```

```
Out[4]= {1.33722 - 0.973846 i, 1.33722 - 0.973846 i}
```

The untruncated Rogers–Ramanujan continued fraction can be expressed in closed form using the function `DedekindEta`. Due to the branch cuts of the logarithm function, this representation does not hold for all q .

```
In[5]:= Rη[q_] =  $\left( \frac{\sqrt{\#^2 + 2\# + 5} - \# - 1}{2} \right) \& \left[ \frac{\text{DedekindEta}\left[-\frac{i \text{Log}[q]}{10\pi}\right]}{\text{DedekindEta}\left[-\frac{i \text{Log}[q^5]}{2\pi}\right]} \right];$ 
```

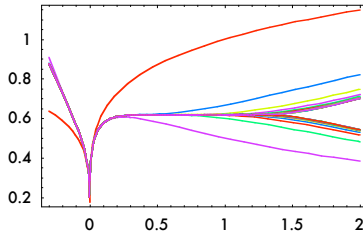
```
In[6]:= With[{q = N[ $\frac{1}{2}$ , 1000]}], Rη[q] - R[q, 200]
```

```
Out[6]= -0. × 10-1001 + 0. × 10-1004 i
```

For any q not too near the unit circle, the truncated continued fraction converges quickly. The following graphic shows the first 36 truncations for real q . While for $0 < q < 1$ the curves converge to one curve, for $q > 1$ we see two curves emerge.

```
In[7]:= With[{o = 36}, Plot[Evaluate[Table[R@R[q, n], {n, 0, o}]],  
  {q, -0.3, 2}, PlotRange → All, Frame → True,  
  Axes → False, PlotStyle → Table[Hue[0.8 n], {n, 0, o}]]
```

From In[7]:=



These two asymptotic values taken by the truncated continued fraction after n divisions $\mathcal{R}_n(q)$ for $|q| > 1$ can be expressed through values inside the unit disk [6]:

$$\mathcal{R}_n(q) = \begin{cases} q^{1/5} (-q^{-1})^{-1/5} \mathcal{R}_n(-q^{-1})^{-1} & n \text{ even} \\ q^{-4/5} (q^{-4})^{-1/5} \mathcal{R}_n(q^{-4}) & n \text{ odd.} \end{cases}$$

The next two inputs confirm this.

```
In[8]:= With[{q = N[ $\pi + e i$ ], n = 100}, {R[q, n], q1/5 (-q-1)1/5 R[-q-1, n]-1}]
```

```
Out[8]= {1.03991 + 0.344971 i, 1.03991 + 0.344971 i}
```

```
In[9]:= With[{q = N[ $\pi + e i$ ], n = 101}, {R[q, n],  $q^{-4/5} (q^{-4})^{-1/5} R[q^{-4}, n]$ }]
Out[9]:= {0.270365 - 0.173169 i, 0.270365 - 0.173169 i}
```

The following infinite product representation for the Rogers–Ramanujan continued fraction strongly suggests that the q -unit circle contains a large set of singularities:

$$\mathcal{R}(q) = q^{1/5} \prod_{k=0}^{\infty} \frac{(1 - q^{5k+1})(1 - q^{5k+4})}{(1 - q^{5k+2})(1 - q^{5k+3})}.$$

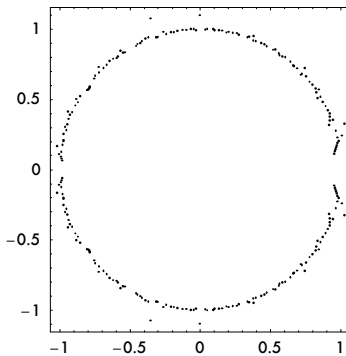
Here is a “togethered” form of a truncated fraction that ignores the $q^{1/5}$ factor.

```
In[10]:= Factor @@ (q-1/5 Together[R[q, 12]])
Out[10]:= ((1 - q + q2 - q3 + q4 - q5 + q6)
(1 + q + q2 + 2 q3 + 2 q4 + 2 q5 + 3 q6 + 3 q7 + 4 q8 + 5 q9 + 5 q10 + 6 q11 +
8 q12 + 8 q13 + 9 q14 + 10 q15 + 10 q16 + 11 q17 + 12 q18 + 11 q19 +
12 q20 + 12 q21 + 11 q22 + 11 q23 + 11 q24 + 10 q25 + 10 q26 + 9 q27 +
8 q28 + 7 q29 + 6 q30 + 4 q31 + 3 q32 + 2 q33 + q34 + q35 + q36)) /
((1 + q + q2 + q3 + q4 + q5 + q6 + q7 + q8 + q9 + q10 + q11 + q12)
(1 + q4 + q6 + q8 + q9 + q10 + q11 + 2 q12 + q13 + q14 + q15 + 2 q16 +
q17 + 2 q18 + q19 + 2 q20 + q21 + 2 q22 + 2 q24 + q25 + q26 + q28 + q30))
```

Next, we truncate after 30 divisions and display the zeros of the resulting denominator, a polynomial of degree 240. The roots clearly cluster around the unit circle, and indeed the unit circle is the natural boundary of analyticity for $\mathcal{R}(q)$.

```
In[11]:= qRoots100 = Solve[Denominator[Together[R[q, 30]]] == 0, q];
In[12]:= Show[Graphics[Point[{Re[#], Im[#]}] & /@ N[q /. qRoots100]],
PlotRange -> All, Frame -> True, AspectRatio -> Automatic]
```

From In[12]:=



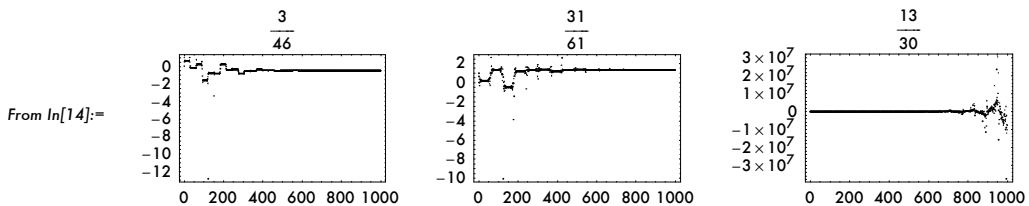
On the unit circle, the continued fraction can converge or diverge [7]. The following graphics show the behavior of the truncated continued fractions for various α in $q = e^{i\pi\alpha}$. To avoid the time-consuming calculation of $\mathcal{R}[q, n]$ for different n , we implement the function `RCFConvergents` that gives all the convergents at once.

```

In[13]:= RCFConvergents[q_, o_] := Module[{a, b, p, q},
  b[n_] := 1; a[n_] = qn-1;
  p[0] = 0; q[0] = 1; p[1] = q1/5; q[1] = 1;
  p[n_] := p[n] = b[n] p[n-1] + a[n] p[n-2];
  q[n_] := q[n] = b[n] q[n-1] + a[n] q[n-2];
  Table[ $\frac{p[k]}{q[k]}$ , {k, 0, o}]]

In[14]:= Off[Graphics::"gptn"];
Show[GraphicsArray[
  ListPlot[Re@RCFConvergents[N[ei π, 300], 1000],
    DisplayFunction → Identity, PlotLabel → #, Axes → False,
    Frame → True, PlotRange → All] & /@ { $\frac{3}{46}$ ,  $\frac{31}{61}$ , (*divergence*)  $\frac{13}{30}$ }]]]

```



```

In[16]:= With[{q = N[1/2, 200]}, {RCFConvergents[q, 100][[-1]] - R[q, 100]}]

```

```

Out[16]= {0. × 10-200}

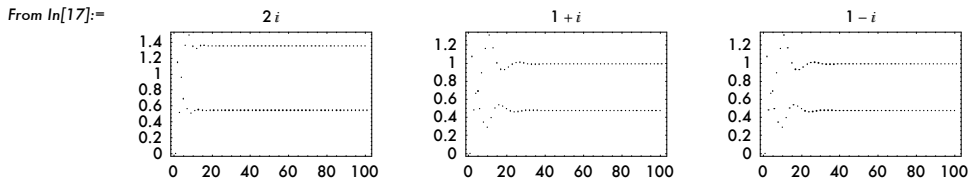
```

Using the function `RCFConvergents` for arguments outside the unit circle shows that the alternating values (known as a two-period) also occurs for complex q -values outside the unit circle.

```

In[17]:= Show[GraphicsArray[
  ListPlot[Abs@RCFConvergents[N[#, 300], 100],
    DisplayFunction → Identity, PlotLabel → #, Axes → False,
    Frame → True, PlotRange → All] & /@ {2 i, 1 + i, 1 - i}]]

```



As a function with a natural boundary at the unit circle, the accumulation of singularities will make a graphic worthwhile. We show contour plots of the real and imaginary parts in the disk $|q| \leq 3/2$.

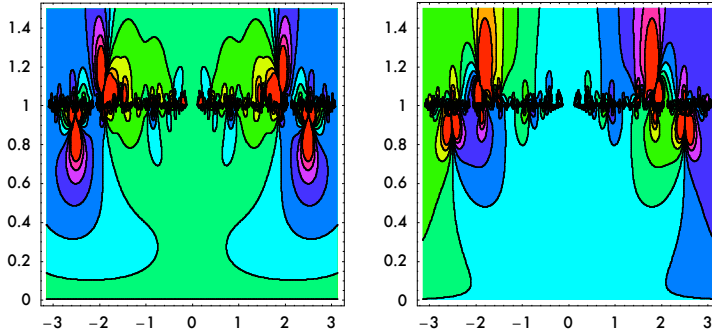
```

In[18]:= ppr = 300; ppφ = 300; ε = 10.-8; rMax = 3/2;
data = Table[R[N[r ei φ], 50],
  {r, ε, rMax, (rMax - ε)/ppr}, {φ, -π + ε, π - ε, (π - 2 ε)/ppφ}];

In[20]:= Show[GraphicsArray[
  ListContourPlot[#data], MeshRange → {{-π + ε, π - ε}, {ε, rMax}},
  DisplayFunction → Identity, ColorFunction → Hue] & /@ {Re, Im}]

```

From In[20]:=



Because a direct call to `ContourPlot` yields relatively poor results, we attempt a homogeneous contour spacing and also map the resulting contour plot on a circular domain. The function `circularContourPlot` does both.

```

In[21]:= toDisk[δ_, Polygon[l_]] :=
Module[{rL, δ1, φD, φ1}, rL = Join@@
  (If[(φD = √[#.# & [Last[#] - First[#]]) < δ, #, δ1 = Last[#] - First[#];
    φ1 = Ceiling[φD/δ]; Table[#[[1] + i δ1/φ1, {i, 0, φ1}]]] & /@
    N[Partition[Append[l, First[l]], 2, 1]]);
  {Polygon[Apply[#2 {Cos[#1], Sin[#1]} &, rL, {1}]]}]

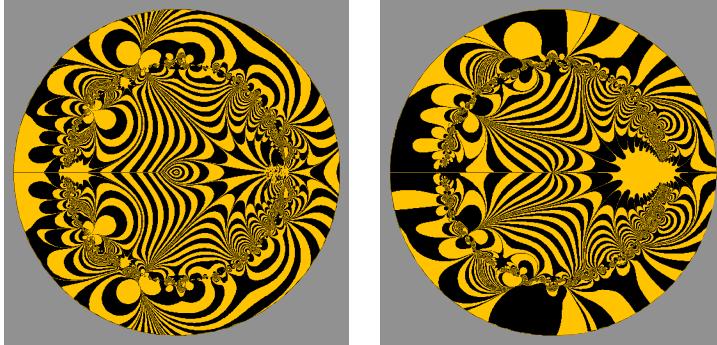
In[22]:= circularContourPlot[data_, cl_] :=
Module[{n = Round[ppr ppφ / cl], m, cls, F, gr, c},
  m = Round[n/2];
  cls = (#[[m]] &) /@ Partition[Sort[Flatten[Re[data]]], n]; {F[c_,
    Which@@Flatten[MapIndexed[{#1, GrayLevel[Mod[First[#2], 2]]} &,
      Append[Function[x, c < x] /@ cls, True]]] // (Set @@ #) &};
  gr = Graphics[ListContourPlot[Re[data], ContourLines → False,
    MeshRange → {{-π + ε, π - ε}, {ε, rMax}}, ColorFunction → F,
    Contours → cls, ColorFunctionScaling → False,
    Frame → False, Background → GrayLevel[1/2]]];
  DeleteCases[gr, DisplayFunction → Identity, ∞] /.
  p_Polygon => toDisk[0.025, p]]

```

The two yellow and black graphics clearly show the clustering of the contour lines due to the dense set of singularities forming the boundary of analyticity at the unit circle.

```
In[23]:= Show[GraphicsArray[(circularContourPlot[# [data], 20] &) /@ {Re, Im}] /.  
          GrayLevel[1] -> Hue[0.12]]
```

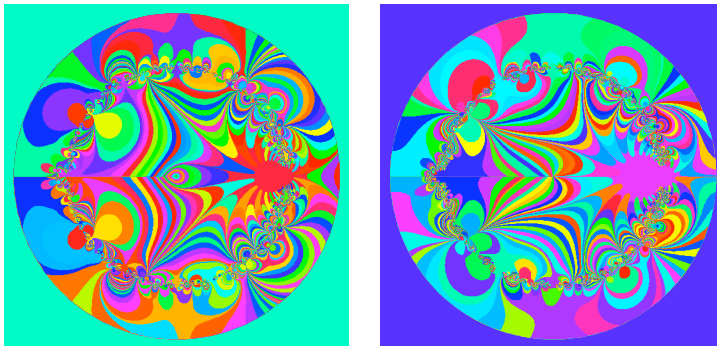
From In[23]:=



We repeat the same calculations for an odd number of divisions to visualize the second value taken asymptotically outside the unit circle. This time we use more psychedelic colors.

```
In[24]:= data = Table[R[N[r e^i *], 51],  
                    {r, ε, rMax, (rMax - ε) / ppr}, {φ, -π + ε, π - ε, (π - 2 ε) / ppφ}];  
  
In[25]:= Show[GraphicsArray[(circularContourPlot[# [data], 20] &) /@ {Re, Im}] /.  
          _GrayLevel -> Hue[Random[]]]
```

From In[25]:=



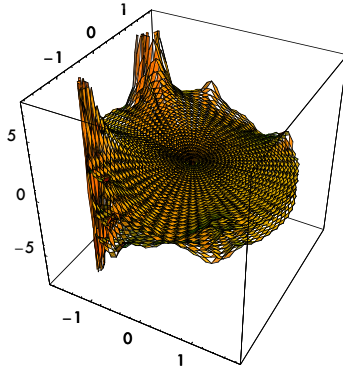
We conclude this section with a Riemann surface (in the spirit of previous Trott's Corners). The factor $q^{1/5}$ makes a nontrivial Riemann surface with five sheets.

```

In[26]:= With[{ppr = 30, ppφ = 60, ε = 10.-8, rMax = 0.9},
  Show[Graphics3D[{SurfaceColor[Hue[0.12], Hue[0.02], 2.4],
    EdgeForm[Thickness[0.001]],
    Cases[Table[ParametricPlot3D[{r Cos[φ], r Sin[φ],
      Re[ej 2 π i/5 R[N[r ei φ], 40]]], {r, ε, rMax}, {φ, -π + ε, π - ε},
      PlotPoints → {ppr, ppφ}, DisplayFunction → Identity],
      {j, 0, 4}], _Polygon, ∞] /. Polygon[l_] =>
      Polygon[Plus @@@ Partition[Append[l, First[l]], 2, 1]]],
    BoxRatios → {1, 1, 1}, PlotRange → {-8, 8}, Axes → True]]

```

From In[26]:=



■ Modular Equations

Now we calculate the simplest modular equations for the Rogers–Ramanujan continued fraction. In the simplest case, modular equations (or identities) are bivariate polynomials $p(\mathcal{R}(q), \mathcal{R}(q^c)) = 0$ for some positive integer c .

Ramanujan found the following one, among others:

$$\frac{\mathcal{R}(q^2) - \mathcal{R}(q)^2}{\mathcal{R}(q^2) + \mathcal{R}(q)^2} = \mathcal{R}(q) \mathcal{R}(q^2)^2.$$

Here is a high-precision check of this identity.

```

In[27]:= q = N[1/π, 200];
u = R[q, 200]; v2 = R[q2, 200];

$$\frac{v_2 - u^2}{v_2 + u^2} - u v_2^2$$


```

Out[29]= $-0. \times 10^{-200}$

In addition to their intrinsic beauty, modular identities are frequently useful for constructing fast algorithms. Repeatedly squaring the argument, $q \rightarrow q^2$, will make it quickly decrease, and a very low approximation of $\mathcal{R}[q, n]$ can be used at the last step. We will not follow this line of thought here and only demonstrate it by using $\mathcal{R}[q, 0]$, which is $q^{1/5}$.

```

In[30]:= Roots[ $\frac{v - u^2}{v + u^2} == u^2, u, \text{Cubics} \rightarrow \text{False}$ ]

Out[30]:= u == Root[-v + v^3 #1 + #1^2 + v^2 #1^3 &, 1] ||
          u == Root[-v + v^3 #1 + #1^2 + v^2 #1^3 &, 2] ||
          u == Root[-v + v^3 #1 + #1^2 + v^2 #1^3 &, 3]

In[31]:= RMEBased[q_,  $\epsilon$ _] := Module[{c = 0, q = q}, While[Abs[q] >  $\epsilon$ , c++; q = q^2];
          Nest[Function[v, Root[(-v #0 + v^3 #1 + #2 + v^2 #3) &, 3]], q^(1/5), c]]

In[32]:= RMEBased[N[ $\frac{1}{2}$ , 200], 10^-80] - R[N[ $\frac{1}{2}$ , 200], 100]

Out[32]:= 1.8685140511145868578497288852633  $\times 10^{-160}$ 

```

The last recursive squaring approach needed only nine steps to yield 134 correct digits for $q = 1/2$. With nine terms, the truncated continued fraction would give only 18 correct digits.

Here is another modular equation for $c = 5$. This time we make a series of order 100 to check the identity. The truncated series vanishes.

```

In[33]:= (v^5 -  $\frac{v(1 - 2v + 4v^2 - 3v^3 + v^4)}{1 + 3v + 4v^2 + 2v^3 + v^4}$ ) /. {u -> R[q, 100], v -> R[q^5, 100]} +
          0[q]^100

Out[33]:= 0[q]^100

```

Now we will implement the few lines of *Mathematica* code that will allow us to generate new modular equations. The idea is similar in spirit to the last check; we will use a high-order q -series: for a fixed c_1 and c_2 , we make the following ansatz:

$$p(\mathcal{R}(q^{c_1}), \mathcal{R}(q^{c_2})) = \sum_{0 \leq i+j \leq d} \alpha_{i,j} \mathcal{R}(q^{c_1})^i \mathcal{R}(q^{c_2})^j.$$

Here $d = d(c_1, c_2)$ is a fixed integer. Expanding $\mathcal{R}(q)$ in a series in q allows a comparison of coefficients, and we get a linear system for the $(d+1)^2$ coefficients $\alpha_{i,j}$. Because of the factor $q^{1/5}$, we can partition the coefficients with respect to their fractional power. The function `splitGroups` does this. We use the abbreviations $u = \mathcal{R}(q^{c_1})$ and $v = \mathcal{R}(q^{c_2})$ to make the code easier to read.

```

In[34]:= splitGroups[d_, {c1_, c2_}, {u_, v_}] :=
          Module[{q}, Sort /@ Map[{#1], #3] &, Split[Sort[
              Join[#, {IntegerPart[Exponent[#2], q]],
                  FractionalPart[Exponent[#2], q]]] & /@
              Transpose[{#, # /. {u -> q^(c1/5), v -> q^(c2/5)}] &][
                  Flatten[Table[u^i v^j, {i, 0, d}, {j, 0, d}]]],
                  #1[-1] < #2[-1] &], #1[-1] == #2[-1] &], {2}]]

```

Here is an example for $c_1 = 1$, $c_2 = 3$, and $d = 4$. We obtain five groups corresponding to $q^{0/5}$, $q^{1/5}$, $q^{2/5}$, $q^{3/5}$, and $q^{4/5}$.

```
In[35]:= groupsd4c13 = splitGroups[4, {1, 3}, {u, v}]
Out[35]:= {{ {1, 0}, {u^2 v, 1}, {u^4 v^2, 2}, {u v^3, 2}, {u^3 v^4, 3}},
  {{u, 0}, {u^3 v, 1}, {v^2, 1}, {u^2 v^3, 2}, {u^4 v^4, 3}},
  {{u^2, 0}, {u^4 v, 1}, {u v^2, 1}, {u^3 v^3, 2}, {u^4, 2}},
  {{u^3, 0}, {v, 0}, {u^2 v^2, 1}, {u^4 v^3, 2}, {u v^4, 2}},
  {{u^4, 0}, {u v, 0}, {u^3 v^2, 1}, {v^3, 1}, {u^2 v^4, 2}}}
```

The function `groupsToSeriesCoefficientLists` uses a high-order series approximation to obtain the linear equations obeyed by the coefficients $\alpha_{i,j}$. To avoid parametrized solutions, we use $\lceil 3/2 \text{ numberOfGroups} \rceil$ series terms to make sure we get more equations than coefficients $\alpha_{i,j}$.

```
In[36]:= groupsToSeriesCoefficientLists[groups_, {c1_, c2_}, {u_, v_}] :=
  With[{n = Ceiling[3/2 Length[groups]]},
    If[Length[#] < n, Join[#, Table[0, {n - Length[#]}]], Take[#, n]] & /@
    CoefficientList[Normal[Series[Apply[#1 q^#2 &, groups, {1}] /.
      {u -> RS[q^c1], v -> RS[q^c2]}, {q, 0, n + 2}]], q]]
In[37]:= RS[q_] = With[{o = 400}, Normal[Series[q^-1/5 R[q, o], {q, 0, o}]]];
```

Here is another example. By using enough series terms, we get (many) more equations than coefficients and hopefully get a unique solution if a modular equation exists.

```
In[38]:= Short[
  groupsToSeriesCoefficientLists[groupsd4c13[[1]], {2, 3}, {u, v}], 6]
Out[38]//Short:= {{1, 0, 0, 0, 0, 0, 0, 0},
  {0, 1, 0, -2, -1, 3, 2, -1}, {0, 0, 1, 0, -4, -2, 10, 8},
  {0, 0, 1, 0, -1, -3, 1, 3}, {0, 0, 0, 1, 0, -3, -4, 6}}}
```

To find nontrivial solutions of the resulting homogeneous system for the $\alpha_{i,j}$, we use the function `NullSpace`. Specifically, we use modular null space computations to avoid intermediate coefficient swell that would arise from using exact rational arithmetic. Using approximate high-precision numbers would lose too much precision in the calculations.

```
In[39]:= makeModularEquationsCandidates[group_, {c1_, c2_},
  {u_, v_}, prime_ : Prime[10^6]] := If[# === {}, {},
  (First /@ group.# &) /@ Mod[#, prime, -Floor[prime/2]]] & [NullSpace[
  Transpose[groupsToSeriesCoefficientLists[group, {c1, c2}, {u, v}]],
  Modulus -> prime]]
```

Here are examples where the system has either no solution or a solution.

```
In[40]:= makeModularEquationsCandidates[groupsd4c13[[1]], {1, 3}, {u, v}]
Out[40]= {}
In[41]:= makeModularEquationsCandidates[groupsd4c13[[4]], {1, 3}, {u, v}]
Out[41]= {-u^3 + v - 3 u^2 v^2 - u^4 v^3 + u v^4}
```

A numerical check confirms that we indeed found a modular equation.

```
In[42]:= % /. {u -> R[N[1/π, 200], 200], v -> R[N[(1/π)^3, 200], 200]}
```

```
Out[42]= {-0. × 10-201}
```

Solving with `Modulus→Prime` guarantees that we do not miss a solution, but such solutions might not be solutions over the rationals. To avoid spurious solutions, we use the function `selectModularEquationNumerically` to select (via a high-precision check) the real modular equations. In addition, `selectModularEquationNumerically` factors the resulting identities and extracts the relevant parts.

```
In[43]:= RN[q_ : 1/π, c_, prec_] := RN[q, c, prec] = R[N[q, prec]^c, prec]
```

```
In[44]:= selectModularEquationNumerically[eqs_, {u_, v_}, {c1_, c2_}, prec_] :=  
  Select[Union[Cases[Factor[Union[Flatten[eqs]], _Plus, ∞]],  
    (# /. {u -> RN[c1, prec], v -> RN[c2, prec]}) == 0 &]
```

Now we have everything to implement the function `findModularEquation` that, for a given c , will find a corresponding modular equation. The function starts with $d = c$ and increases d until a modular equation is found. The optional second argument allows us to start the search at a given value of d and a given group number j .

```
In[45]:= findModularEquation[{c1_, c2_}, d0j0_ : {0, 1}] :=  
  Module[{d = d0j0[[1]], groups, candidates, mEqs = {}},  
    While[groups = splitGroups[Max[c1, c2] + d, {c1, c2}, {u, v}];  
      j = If[d == d0j0[[1]], d0j0[[2]], 1];  
      While[j <= Length[groups] ∧ (candidates =  
        makeModularEquationsCandidates[groups[[j]], {c1, c2}, {u, v}];  
        mEqs = selectModularEquationNumerically[candidates, {u, v},  
          {c1, c2}, 100]; mEqs == {}), j++]; mEqs == {}, d++]; mEqs]
```

```
In[46]:= findModularEquation[c_, d0j0_ : {0, 1}] :=  
  findModularEquation[{1, c}, d0j0]
```

The next input recalculates the modular equation given earlier.

```
In[47]:= findModularEquation[2]
```

```
Out[47]= {u2 - v + u3 v2 + u v3}
```

The following input avoids the search.

```
In[48]:= findModularEquation[3, {1, 4}]
```

```
Out[48]= {-u3 + v - 3 u2 v2 - u4 v3 + u v4}
```

Here is a complete list of two-variable modular equations for $c_1 = 1$, $2 \leq (c = c_2) \leq 15$ (only the equations for $c = 2, 3, 4, 5, 7, 11$ were known [8, 9, 10]). For brevity of the results, we apply `Simplify`. We use the notations $u = R[q]$, and $u = R[q^c]$.

```
In[49]:= Apply[(CellPrint[Cell[TextData[{"Modular equation of order ",
      Cell[BoxData[FormBox[ToString[#1], TraditionalForm]]],
      ":"}], "Text"]]; Print[
      TraditionalForm[Simplify[findModularEquation[##][[1]] == 0]] &,
      Take[{{2, {1, 3}}, {3, {1, 4}}, {4, {2, 1}}, {5, {0, 1}},
        {6, {6, 3}}, {7, {1, 3}}, {8, {4, 4}}, {9, {3, 1}},
        {10, {5, 1}}, {11, {1, 3}}, {12, {12, 2}},
        {13, {1, 4}}, {14, {10, 1}}, {15, {5, 1}}}, All], {1}];
```

Modular equation of order 2:

$$v^2 u^3 + u^2 + v^3 u - v = 0$$

Modular equation of order 3:

$$-v^3 u^4 - u^3 - 3 v^2 u^2 + v^4 u + v = 0$$

Modular equation of order 4:

$$v u^6 + (v^5 - 1) u^5 - 5 v^4 u^4 + 10 v^3 u^3 - 5 v^2 u^2 + (v^6 + v) u - v^5 = 0$$

Modular equation of order 5:

$$v(v^4 - 3 v^3 + 4 v^2 - 2 v + 1) - u^5(v^4 + 2 v^3 + 4 v^2 + 3 v + 1) = 0$$

Modular equation of order 6:

$$\begin{aligned} &v^5 u^{12} + v^6 (v^5 + 6) u^{11} + v^2 (9 v^5 - 1) u^{10} - 5 (v^8 + v^3) u^9 - 5 v^9 u^8 + \\ &(9 v^{10} + 36 v^5 - 1) u^7 + v (6 v^{10} + 59 v^5 - 6) u^6 + v^2 (v^{10} + 36 v^5 - 9) u^5 + \\ &5 v^3 u^4 - 5 v^4 (v^5 - 1) u^3 - v^5 (v^5 + 9) u^2 + (v - 6 v^6) u - v^7 = 0 \end{aligned}$$

Modular equation of order 7:

$$\begin{aligned} &-v^7 u^8 + (7 v^5 - 1) u^7 - 7 v^3 u^6 + 7 v (v^5 - 1) u^5 - \\ &35 v^4 u^4 - 7 (v^7 + v^2) u^3 + 7 v^5 u^2 + v^3 (v^5 + 7) u + v = 0 \end{aligned}$$

Modular equation of order 8:

$$\begin{aligned} &-v^9 u^{12} + v (v^{10} + 8 v^5 - 1) u^{11} - 3 (3 v^8 + v^3) u^{10} + \\ &(3 v^{10} - 18 v^5 - 1) u^9 - 9 (2 v^7 + v^2) u^8 + 18 v^4 (v^5 - 1) u^7 + \\ &2 v (4 v^{10} + 65 v^5 - 4) u^6 + 18 (v^8 + v^3) u^5 - 9 v^5 (v^5 - 2) u^4 + \\ &v^2 (v^{10} - 18 v^5 - 3) u^3 + 3 v^4 (v^5 - 3) u^2 + (-v^{11} - 8 v^6 + v) u + v^3 = 0 \end{aligned}$$

Modular equation of order 9:

$$\begin{aligned} &v^2 u^{12} - (v^{11} + 9 v^6 + v) u^{11} + (v^{10} - 12 v^5 + 1) u^{10} - \\ &10 v^4 (v^5 - 3) u^9 + 15 v^3 (3 v^5 - 2) u^8 - 12 v^2 (9 v^5 - 1) u^7 - \\ &9 v (v^{10} - 3 v^5 - 1) u^6 - 12 v^5 (v^5 + 9) u^5 + 15 v^4 (2 v^5 + 3) u^4 - \\ &10 (3 v^8 + v^3) u^3 + (v^{12} + 12 v^7 + v^2) u^2 - (v^{11} - 9 v^6 + v) u + v^{10} = 0 \end{aligned}$$

Modular equation of order 10:

$$\begin{aligned} &v^2 (v^4 - 3v^3 + 4v^2 - 2v + 1)^2 (v^4 + 2v^3 + 4v^2 + 3v + 1)u^{15} + \\ &(10v^{14} - 50v^{13} + 81v^{12} - 184v^{11} + 274v^{10} - 275v^9 + \\ &285v^8 - 54v^7 + 31v^6 + 14v^5 + 85v^4 + 5v^3 + 9v^2 - v + 1)u^{10} + \\ &v(v^{14} + v^{13} + 9v^{12} - 5v^{11} + 85v^{10} - 14v^9 + 31v^8 + 54v^7 + 285v^6 + \\ &275v^5 + 274v^4 + 184v^3 + 81v^2 + 50v + 10)u^5 - \\ &v(v^4 - 3v^3 + 4v^2 - 2v + 1)(v^4 + 2v^3 + 4v^2 + 3v + 1)^2 = 0 \end{aligned}$$

Modular equation of order 11:

$$\begin{aligned} &u^{12} - v(v^{10} + 11v^5 + 11)u^{11} + 66v^2u^{10} - 220v^3u^9 + \\ &495v^4u^8 - 792v^5u^7 - 11v(v^{10} - 73v^5 - 1)u^6 - 792v^7u^5 + \\ &495v^8u^4 - 220v^9u^3 + 66v^{10}u^2 - (11v^{11} - 11v^6 + v)u + v^{12} = 0 \end{aligned}$$

Modular equation of order 12:

$$\begin{aligned} &v^{16}u^{24} + (v^{19} + 18v^{14} - 12v^9 + v^4)u^{23} + v^7(15v^{10} + 105v^5 - 7)u^{22} + \\ &v^5(v^{15} + 74v^{10} + 220v^5 - 15)u^{21} + v^3(v^{20} + 29v^{15} + 2v^{10} + 91v^5 - 1)u^{20} + \\ &v(15v^{20} + 259v^{15} - 100v^{10} + 85v^5 - 1)u^{19} + \\ &v^4(85v^{15} + 430v^{10} - 40v^5 + 29)u^{18} + \\ &v^2(7v^{20} + 245v^{15} + 490v^{10} - 245v^5 + 15)u^{17} + \\ &(91v^{20} + 266v^{15} + 1316v^{10} - 259v^5 + 1)u^{16} + \\ &2v^3(6v^{20} + 20v^{15} - 667v^{10} + 133v^5 - 37)u^{15} + \\ &2v(110v^{20} + 658v^{15} - 1221v^{10} + 215v^5 - 9)u^{14} + \\ &2(50v^{19} + 1221v^{14} + 667v^9 + v^4)u^{13} + \\ &7v^2(15v^{20} + 70v^{15} - 347v^{10} - 70v^5 + 15)u^{12} - \\ &2v^5(v^{15} - 667v^{10} + 1221v^5 - 50)u^{11} + \\ &2v^3(9v^{20} + 215v^{15} + 1221v^{10} + 658v^5 - 110)u^{10} - \\ &2v(37v^{20} + 133v^{15} + 667v^{10} + 20v^5 - 6)u^9 + \\ &v^4(v^{20} + 259v^{15} + 1316v^{10} - 266v^5 + 91)u^8 - \\ &v^2(15v^{20} + 245v^{15} + 490v^{10} - 245v^5 + 7)u^7 + \\ &v^5(29v^{15} + 40v^{10} + 430v^5 - 85)u^6 - \\ &v^3(v^{20} + 85v^{15} + 100v^{10} + 259v^5 - 15)u^5 + \\ &v(v^{20} + 91v^{15} - 2v^{10} + 29v^5 - 1)u^4 - (15v^{19} + 220v^{14} - 74v^9 + v^4)u^3 + \\ &v^7(7v^{10} + 105v^5 - 15)u^2 - v^5(v^{15} + 12v^{10} + 18v^5 - 1)u + v^8 = 0 \end{aligned}$$

Modular equation of order 13:

$$\begin{aligned} &-v^{13}u^{14} + (-26v^{10} + 13v^5 - 1)u^{13} + 13(v^{12} - 12v^7 + v^2)u^{12} + \\ &26v^4(5v^5 - 1)u^{11} - 13v(2v^{10} + 33v^5 - 2)u^{10} - 13v^3(v^{10} - 33v^5 + 10)u^9 - \\ &429(v^{10} + v^5)u^8 + 156v^2(v^{10} - 1)u^7 + 429v^4(v^5 - 1)u^6 - \\ &13(10v^{11} + 33v^6 + v)u^5 - 13v^3(2v^{10} + 33v^5 - 2)u^4 + \\ &26v^5(v^5 + 5)u^3 + 13(v^{12} + 12v^7 + v^2)u^2 + v^4(v^{10} + 13v^5 + 26)u + v = 0 \end{aligned}$$

Modular equation of order 14:

$$\begin{aligned}
 &v^9 u^{24} + v^{13} (v^{10} + 14v^5 + 35)u^{23} + v^2 (14v^{20} + 161v^{15} + 28v^{10} + 7v^5 - 1)u^{22} + \\
 &v^6 (70v^{15} + 771v^{10} + 210v^5 - 35)u^{21} + \\
 &7v^5 (28v^{15} + 326v^{10} - v^5 - 3)u^{20} + 21 (25v^{19} + 263v^{14} - 27v^9 + v^4)u^{19} + \\
 &7v^3 (2v^{20} + 154v^{15} + 1441v^{10} - 180v^5 + 5)u^{18} + \\
 &v^2 (161v^{20} + 2618v^{15} + 10990v^{10} - 1112v^5 - 7)u^{17} + \\
 &3v^6 (257v^{15} + 3367v^{10} + 4291v^5 - 420)u^{16} + \\
 &(2282v^{20} + 25907v^{15} + 5509v^{10} - 567v^5 - 1)u^{15} + \\
 &v^4 (5523v^{15} + 59011v^{10} - 5509v^5 - 7)u^{14} + \\
 &7v^3 (5v^{20} + 1441v^{15} + 15469v^{10} - 1839v^5 + 30)u^{13} + \\
 &7v^2 (4v^{20} + 1570v^{15} + 17259v^{10} - 1570v^5 + 4)u^{12} + \\
 &7v (30v^{20} + 1839v^{15} + 15469v^{10} - 1441v^5 + 5)u^{11} + \\
 &v^5 (-7v^{15} + 5509v^{10} + 59011v^5 - 5523)u^{10} + \\
 &v^4 (v^{20} - 567v^{15} - 5509v^{10} + 25907v^5 - 2282)u^9 - \\
 &3v^3 (420v^{15} + 4291v^{10} - 3367v^5 + 257)u^8 + \\
 &v^2 (7v^{20} - 1112v^{15} - 10990v^{10} + 2618v^5 - 161)u^7 - \\
 &7v (5v^{20} + 180v^{15} + 1441v^{10} - 154v^5 + 2)u^6 - \\
 &21v^5 (v^{15} + 27v^{10} + 263v^5 - 25)u^5 + 7v^4 (3v^{15} - v^{10} - 326v^5 + 28)u^4 + \\
 &v^3 (35v^{15} + 210v^{10} - 771v^5 + 70)u^3 - \\
 &v^2 (v^{20} + 7v^{15} - 28v^{10} + 161v^5 - 14)u^2 + (35v^{11} - 14v^6 + v)u - v^{15} = 0
 \end{aligned}$$

Modular equation of order 15:

$$\begin{aligned}
 &-v^3 (v^4 - 3v^3 + 4v^2 - 2v + 1)^3 (v^4 + 2v^3 + 4v^2 + 3v + 1)u^{20} + \\
 &(-15v^{19} + 60v^{18} - 270v^{17} + 719v^{16} - 1133v^{15} + 2075v^{14} - \\
 &2750v^{13} + 2590v^{12} - 1919v^{11} + 1163v^{10} - 1045v^9 + 380v^8 + \\
 &30v^7 + 23v^6 - 231v^5 + 35v^4 + 10v^3 - 10v^2 + 3v - 1)u^{15} - \\
 &3v (15v^{18} + 71v^{17} + 8v^{16} - 65v^{15} - 185v^{14} + 505v^{13} + 514v^{12} + \\
 &567v^{11} - 1525v^{10} - 660v^9 + 1525v^8 + 567v^7 - \\
 &514v^6 + 505v^5 + 185v^4 - 65v^3 - 8v^2 + 71v - 15)u^{10} + \\
 &v (v^{19} + 3v^{18} + 10v^{17} + 10v^{16} - 35v^{15} - 231v^{14} - 23v^{13} + 30v^{12} - \\
 &380v^{11} - 1045v^{10} - 1163v^9 - 1919v^8 - 2590v^7 - 2750v^6 - \\
 &2075v^5 - 1133v^4 - 719v^3 - 270v^2 - 60v - 15)u^5 + \\
 &v (v^4 - 3v^3 + 4v^2 - 2v + 1) (v^4 + 2v^3 + 4v^2 + 3v + 1)^3 = 0
 \end{aligned}$$

Calculating modular equations for even larger c can be done quickly. Here are the three primes $c = 17$, $c = 19$, and $c = 23$. Primes are often of especial interest for modular equations.

In[50]:= **findModularEquation[17, {1, 3}] // Timing // Simplify**

Out[50]= {79.71Second, { $v - u^{18} v^{17} + 2210 u^9 v^4 (-1 + v^{10}) -$
 $17 u^2 v^5 (-16 + 21 v^5 + 2 v^{10}) + 17 u^6 v^3 (-5 - 309 v^5 + 5 v^{10}) +$
 $17 u^{12} v^5 (-5 - 309 v^5 + 5 v^{10}) - 17 u^{16} v^3 (-2 + 21 v^5 + 16 v^{10}) -$
 $17 u^7 v^5 (149 + 337 v^5 + 59 v^{10}) - 17 u^{11} v^3 (59 - 337 v^5 + 149 v^{10}) +$
 $170 u^{14} (v^4 - 13 v^9 + v^{14}) + 170 u^4 (v^4 + 13 v^9 + v^{14}) -$
 $17 u^{13} v^2 (16 - 149 v^5 - 5 v^{10} + v^{15}) - 17 u^3 v^2 (-2 - 59 v^5 + 5 v^{10} + v^{15}) +$
 $u v^3 (17 + 68 v^5 + 17 v^{10} + v^{15}) - 17 u^8 v^2 (21 + 337 v^5 + 309 v^{10} + 4 v^{15}) +$
 $17 u^5 v (-1 - 5 v^5 + 149 v^{10} + 16 v^{15}) + u^{17} (-1 + 17 v^5 - 68 v^{10} + 17 v^{15}) -$
 $17 u^{10} v (-4 + 309 v^5 - 337 v^{10} + 21 v^{15}) - 17 u^{15} (v - 5 v^6 - 59 v^{11} + 2 v^{16}))$ }}

In[51]:= **findModularEquation[19, {1, 1}] // Timing // Simplify**

Out[51]= {101.9Second,
 $\{u^{20} + v^{20} - 38 u^2 v^7 (4 - 45 v^5 + 4 v^{10}) + 38 u^{18} v^3 (4 + 45 v^5 + 4 v^{10}) -$
 $114 u^{15} v^5 (125 + 121 v^5 + 11 v^{10}) - 418 u^{10} v^5 (-33 - 79 v^5 + 33 v^{10}) -$
 $114 u^5 v^5 (11 - 121 v^5 + 125 v^{10}) - 19 u^{14} v^4 (-608 - 1881 v^5 - 197 v^{10} + v^{15}) -$
 $76 u^{17} v^2 (2 + 79 v^5 + 18 v^{10} + v^{15}) - u^{19} v^4 (76 + 95 v^5 + 19 v^{10} + v^{15}) +$
 $76 u^{13} v^3 (-79 - 765 v^5 - 103 v^{10} + 2 v^{15}) +$
 $76 u^3 v^3 (-1 + 18 v^5 - 79 v^{10} + 2 v^{15}) - 19 u^4 v^4 (-11 + 168 v^5 - 608 v^{10} + 4 v^{15}) -$
 $19 u^9 v^4 (168 + 1683 v^5 - 1881 v^{10} + 5 v^{15}) +$
 $19 u^{16} v (4 + 608 v^5 + 168 v^{10} + 11 v^{15}) - 114 u^{12} v^2$
 $(-15 - 510 v^5 - 74 v^{10} + 12 v^{15}) + 114 u^8 v^3 (12 + 74 v^5 - 510 v^{10} + 15 v^{15}) +$
 $u v (-1 + 19 v^5 - 95 v^{10} + 76 v^{15}) - 76 u^7 v^2 (2 + 103 v^5 - 765 v^{10} + 79 v^{15}) +$
 $19 u^{11} v (-5 - 1881 v^5 - 1683 v^{10} + 168 v^{15}) +$
 $19 u^6 (v + 197 v^6 - 1881 v^{11} + 608 v^{16}))$ }}

In[52]:= **findModularEquation[23, {1, 4}] // Timing // Simplify**

Out[52]= {195.04Second, { $v - u^{24} v^{23} + 253 u^3 v^5 (5 + 21 v^5 + 24 v^{10} + 2 v^{15}) +$
 $253 u^{18} v^5 (30 - 335 v^5 - 108 v^{10} + 5 v^{15}) +$
 $253 u^{21} v^4 (-2 + 24 v^5 - 21 v^{10} + 5 v^{15}) + 759 u^{16} v^4$
 $(14 - 168 v^5 - 183 v^{10} + 14 v^{15}) + 759 u^8 v^5 (-14 - 183 v^5 + 168 v^{10} + 14 v^{15}) -$
 $253 u^6 v^4 (-5 - 108 v^5 + 335 v^{10} + 30 v^{15}) +$
 $46 u^{13} v^5 (-776 + 1362 v^5 - 3967 v^{10} + 391 v^{15}) -$
 $46 u^{11} v^4 (391 + 3967 v^5 + 1362 v^{10} + 776 v^{15}) +$
 $u^{23} (-1 + 23 v^5 - 161 v^{10} + 322 v^{15} - 23 v^{20}) -$
 $23 u^4 v^3 (-22 - 462 v^5 - 782 v^{10} - 55 v^{15} + v^{20}) +$
 $u v^4 (23 + 322 v^5 + 161 v^{10} + 23 v^{15} + v^{20}) -$
 $23 u^{19} v^3 (55 - 462 v^5 - 1552 v^{10} + 330 v^{15} + v^{20}) -$
 $46 u^9 v^3 (132 + 2772 v^5 - 3967 v^{10} - 594 v^{15} + 7 v^{20}) -$
 $23 u^{14} v^3 (-231 + 6039 v^5 - 2724 v^{10} + 3685 v^{15} + 7 v^{20}) -$
 $23 u^{20} v (-1 - 55 v^5 + 782 v^{10} - 462 v^{15} + 22 v^{20}) -$
 $161 u^{12} v^2 (32 + 11 v^5 + 4341 v^{10} - 11 v^{15} + 32 v^{20}) -$
 $23 u^{17} v^2 (89 - 869 v^5 - 77 v^{10} + 869 v^{15} + 32 v^{20}) +$
 $23 u^7 v^2 (32 - 869 v^5 - 77 v^{10} + 869 v^{15} + 89 v^{20}) -$
 $46 u^{15} v (7 + 594 v^5 - 3967 v^{10} - 2772 v^{15} + 132 v^{20}) -$
 $23 u^{10} v (-7 + 3685 v^5 + 2724 v^{10} + 6039 v^{15} + 231 v^{20}) -$
 $23 u^5 (v - 330 v^6 - 1552 v^{11} + 462 v^{16} + 55 v^{21}) -$
 $23 u^2 (v^2 - 89 v^7 + 224 v^{12} + 32 v^{17} + v^{22}) -$
 $23 u^{22} (v^2 - 32 v^7 + 224 v^{12} + 89 v^{17} + v^{22}))$ }}

We check the last identity to 10,000 digits at a “random” q -value.

$$\text{In}[53]:= \% \Pi[2] /. \text{With}\left[\left\{q = N\left[\frac{1}{e^\pi + \pi^e}, 10000\right]\right\}, \{u \rightarrow R[q, 500], v \rightarrow R[q^{23}, 500]\}\right]$$

$$\text{Out}[53]= \{-0. \times 10^{-10007}\}$$

Combining modular equations coming from different values of c allows us to derive new ones. For example, we replace q by q^3 in the modular equation for $c = 6$. By using the modular equation for $c = 3$, we can eliminate the $R(q^3)$. So we obtain the modular equation for $c = 18$. This complicated polynomial that corresponds to $d = 48$ would take a long time to find using the function `findModularEquation` directly.

$$\begin{aligned} \text{In}[54]:= & \text{selectModularEquationNumerically}[\\ & \{(\text{Resultant}[\#, R[q^3]] \& \text{@@}\{\text{findModularEquation}[3, \{1, 4\}]\Pi[1] /. \\ & \{u \rightarrow R[q], v \rightarrow R[q^3]\}, \text{findModularEquation}[6, \{6, 3\}]\Pi[1] /. \\ & \{u \rightarrow R[q], v \rightarrow R[q^6]\} /. q \rightarrow q^3)\} /. \\ & \{R[q] \rightarrow u, R[q^{18}] \rightarrow v\}, \{u, v\}, \{1, 18\}, 100] // \text{Simplify} \\ \text{Out}[54]= & \{v^{14} + u^{36} v^{22} - u v^{12} (-1 + 42 v^5 + 81 v^{10} + 18 v^{15} + v^{20}) + \\ & u^{35} (v^4 - 18 v^9 + 81 v^{14} - 42 v^{19} - v^{24}) + \\ & u^{34} v^6 (-19 + 144 v^5 + 756 v^{10} + 25 v^{15} + 10 v^{20} + v^{25}) + \\ & 2 u^7 v^5 (302 - 4743 v^5 + 20616 v^{10} - 56243 v^{15} - 4583 v^{20} + 46 v^{25}) + \\ & 2 u^{29} v^6 (-46 - 4583 v^5 + 56243 v^{10} + 20616 v^{15} + 4743 v^{20} + 302 v^{25}) + \\ & u^{12} v^5 (-8624 + 84291 v^5 - 25698 v^{10} + 1081806 v^{15} + 90283 v^{20} + 46 v^{25} - v^{30}) + \\ & u^3 v^3 (-1 + 51 v^5 - 318 v^{10} - 723 v^{15} - 2560 v^{20} - 217 v^{25} + v^{30}) + \\ & u^{32} v^5 (21 - 330 v^5 + 841 v^{10} + 7039 v^{15} + 941 v^{20} + 34 v^{25} + v^{30}) - \\ & u^5 v^4 (-21 + 156 v^5 - 794 v^{10} + 19216 v^{15} + 8624 v^{20} + 604 v^{25} + v^{30}) + \\ & u^{10} v^4 (-330 - 6777 v^5 + 144884 v^{10} - 162033 v^{15} + 84291 v^{20} + 9486 v^{25} + 10 v^{30}) + \\ & u^{27} v^5 (156 + 6777 v^5 - 106666 v^{10} - 45221 v^{15} - 19722 v^{20} - 1299 v^{25} + 18 v^{30}) - \\ & u^2 (v^5 - 10 v^{10} + 25 v^{15} - 756 v^{20} + 144 v^{25} + 19 v^{30}) + \\ & u^6 v^2 (-19 + 92 v^5 + 46 v^{10} + 21579 v^{15} - 15021 v^{20} - 1299 v^{25} + 34 v^{30}) + \\ & u^{17} v^5 (19216 + 162033 v^5 - 4324314 v^{10} + \\ & 656403 v^{15} + 348224 v^{20} + 21579 v^{25} + 42 v^{30}) + \\ & u^9 v (18 + 1299 v^5 - 19722 v^{10} + 45221 v^{15} - 106666 v^{20} - 6777 v^{25} + 156 v^{30}) + 2 u^{13} \\ & v^3 (1280 - 5399 v^5 - 131828 v^{10} + 341041 v^{15} + 131828 v^{20} + 10179 v^{25} + 159 v^{30}) + \\ & u^{28} v^3 (51 + 978 v^5 - 10798 v^{10} - 88317 v^{15} - 20358 v^{20} + 978 v^{25} + 217 v^{30}) + u^{18} v^3 \\ & (-723 - 88317 v^5 + 682082 v^{10} + 4638861 v^{15} - 682082 v^{20} - 88317 v^{25} + 723 v^{30}) + \\ & u^{20} v^4 (7039 + 45221 v^5 - 1314056 v^{10} - 656403 v^{15} + \\ & 1081806 v^{20} + 112486 v^{25} + 756 v^{30}) + 2 u^{23} v^3 \\ & (159 - 10179 v^5 + 131828 v^{10} - 341041 v^{15} - 131828 v^{20} + 5399 v^{25} + 1280 v^{30}) + \\ & u^{16} v^2 (756 - 112486 v^5 + 1081806 v^{10} + 656403 v^{15} - 1314056 v^{20} - \\ & 45221 v^{25} + 7039 v^{30}) + u^{19} v (42 - 21579 v^5 + 348224 v^{10} - \\ & 656403 v^{15} - 4324314 v^{20} - 162033 v^{25} + 19216 v^{30}) - \\ & u^4 (v - 34 v^6 + 941 v^{11} - 7039 v^{16} + 841 v^{21} + 330 v^{26} + 21 v^{31}) + \\ & u^{24} (v + 46 v^6 - 90283 v^{11} + 1081806 v^{16} + 25698 v^{21} + 84291 v^{26} + 8624 v^{31}) + \\ & u^{11} (144 v^2 + 9166 v^7 - 90283 v^{12} - 348224 v^{17} + 294351 v^{22} + 19722 v^{27} - 941 v^{32}) + \\ & u^{21} (25 v^2 - 41232 v^7 + 25698 v^{12} + \\ & 4324314 v^{17} + 1646062 v^{22} + 106666 v^{27} - 841 v^{32}) + \\ & u^{26} (10 v^2 - 9486 v^7 + 84291 v^{12} + 162033 v^{17} + 144884 v^{22} + 6777 v^{27} - 330 v^{32}) + \\ & u^{31} (v^2 - 604 v^7 + 8624 v^{12} - 19216 v^{17} - 794 v^{22} - 156 v^{27} - 21 v^{32}) + \end{aligned}$$

$$\begin{aligned}
& u^8 (-217 v^3 + 978 v^8 + 20358 v^{13} - 88317 v^{18} + 10798 v^{23} + 978 v^{28} - 51 v^{33}) + \\
& u^{33} (v^3 + 217 v^8 - 2560 v^{13} + 723 v^{18} - 318 v^{23} - 51 v^{28} - v^{33}) + \\
& u^{25} (941 v^4 + 19722 v^9 - 294351 v^{14} - 348224 v^{19} + 90283 v^{24} + 9166 v^{29} - 144 v^{34}) + \\
& u^{15} (841 v^4 + 106666 v^9 - 1646062 v^{14} + \\
& \quad 4324314 v^{19} - 25698 v^{24} - 41232 v^{29} - 25 v^{34}) + \\
& u^{30} (34 v^4 + 1299 v^9 - 15021 v^{14} - 21579 v^{19} + 46 v^{24} - 92 v^{29} - 19 v^{34}) + \\
& u^{14} v (-81 - 15021 v^5 + 294351 v^{10} - 1314056 v^{15} + 1646062 v^{20} + \\
& \quad 144884 v^{25} - 794 v^{30} + v^{35}) + u^{22} (1 + 794 v^5 + 144884 v^{10} - \\
& \quad 1646062 v^{15} - 1314056 v^{20} - 294351 v^{25} - 15021 v^{30} + 81 v^{35}) \}
\end{aligned}$$

$$\text{In}[55]:= \% /. \text{With}\left[\left\{q = \mathbf{N}\left[\frac{1}{\pi^{\pi} + \mathbf{e}^{\mathbf{e}}}, 10000\right]\right\}, \{u \rightarrow \mathcal{R}[q, 500], v \rightarrow \mathcal{R}[q^{18}, 500]\}\right]$$

$$\text{Out}[55]= \{-0. \times 10^{-10007}\}$$

Using `findModularEquation` using a list as the first argument allows us to find modular equations of the form $p(\mathcal{R}(q^{c_1}), \mathcal{R}(q^{c_2}))$. Here is a simple example.

$$\text{In}[56]:= \text{findModularEquation}[\{2, 3\}]$$

$$\begin{aligned}
\text{Out}[56]= \{-u^5 + 6 u^6 v + u^{11} v + u^2 v^2 - 9 u^7 v^2 - 5 u^3 v^3 - 5 u^8 v^3 + 5 u^9 v^4 - v^5 + \\
36 u^5 v^5 + 9 u^{10} v^5 + 6 u v^6 - 59 u^6 v^6 - 6 u^{11} v^6 - 9 u^2 v^7 + 36 u^7 v^7 + u^{12} v^7 - \\
5 u^3 v^8 + 5 u^4 v^9 - 5 u^9 v^9 + 9 u^5 v^{10} + u^{10} v^{10} + u v^{11} - 6 u^6 v^{11} + u^7 v^{12}\}
\end{aligned}$$

This concludes our short excursion into the world of modular equations. The interested reader could now go on and search for more complicated modular equations, such as $p(\mathcal{R}(q^{c_1}), \mathcal{R}(q^{c_2}), \mathcal{R}(q^{c_3})) = 0$ for positive integers c_1, c_2 , and c_3 like the following two examples for $p(\mathcal{R}(q^1), \mathcal{R}(q^2), \mathcal{R}(q^3))$, and $p(\mathcal{R}(q^2), \mathcal{R}(q^3), \mathcal{R}(q^5))$ (where we use $w = \mathcal{R}(q^{c_3})$).

$$\text{In}[57]:= \text{tripleME123} = v w + u^4 (-v^3 + v w^3) + u^3 v^2 w (-1 + v w (-3 + v w)) + u^2 w (-1 + v^2 w^2 (-1 + v w));$$

$$\begin{aligned}
\text{In}[58]:= \text{tripleME235} = 10 u^7 v^4 w (1 + (-1 + w) w (2 + (-2 + w) w)) + \\
10 u^2 v^4 (1 + w (1 + w) (3 + w + w^2)) + u^8 (w + (-1 + w) w^2 (2 + (-2 + w) w) + \\
v^5 (1 - w (1 + (-1 + w) w) (3 + w (-13 + 6 w)))) + \\
u^9 v (1 + w ((1 + w) (3 + w + w^2) + v^5 (1 + (-1 + w) w (2 + (-2 + w) w)))) + \\
v^2 (1 + w ((1 + w) (3 + w + w^2) + v^5 (1 + (-1 + w) w (2 + (-2 + w) w)))) - \\
u v^3 (6 + v^5 (1 + w (1 + w) (3 + w + w^2)) + w (19 + w (22 + w (16 + w (3 + w)))) + \\
u^4 v (5 + w (14 + w (22 + w (6 - (-8 + w) w))) + \\
6 v^5 (-1 + w (-2 + w (-6 + w (2 + (-4 + w) w)))) + \\
u^6 v^3 (-3 + w (-8 + w (-14 + w (-2 + (-6 + w) w))) + \\
v^5 (1 + w (4 + w (2 + w (6 + (-2 + w) w)))) + \\
u^3 (-1 + w (-2 + w (-6 + w (2 + (-4 + w) w))) + \\
v^5 (1 + w (6 + w (-2 + w (14 + w (-8 + 3 w)))))) - \\
u^5 v^2 (6 (1 + w (4 + w (2 + w (6 + (-2 + w) w)))) + \\
v^5 (1 + w (8 + w (-6 + w (22 + w (-14 + 5 w))))));
\end{aligned}$$

$$\begin{aligned}
\text{In}[59]:= \text{checkNumerically}[\text{expr}_-, \{c1_ , c2_ , c3_ \}] := \\
\text{expr} /. \text{With}\left[\left\{q = \mathbf{N}\left[\frac{\pi^{\mathbf{e}}}{2 \mathbf{e}^{\pi}}, 1000\right]\right\}, \right. \\
\left. \{u \rightarrow \mathcal{RN}[q, c1, 500], v \rightarrow \mathcal{RN}[q, c2, 500], w \rightarrow \mathcal{RN}[q, c3, 500]\}\right]
\end{aligned}$$

```
In[60]:= {checkNumerically[tripleME123, {1, 2, 3}],
          checkNumerically[tripleME235, {2, 3, 5}]}
Out[60]= {0. × 10-500, -0. × 10-499}
```

■ Exact Values

In conclusion let us take a quick look at another fascinating facet of the Rogers–Ramanujan continued fractions. Ramanujan observed that, interestingly, $\mathcal{R}(e^{-\pi\sqrt{r}})$ is an algebraic number for positive rational r . The most celebrated instance is $\mathcal{R}(e^{-2\pi})$. The function `Recognize` from the `NumberTheory`` package allows us to identify an algebraic number given from an approximate value.

```
In[61]:= Needs["NumberTheory`Recognize`"]
In[62]:= r = R[N[e-2π, 100], 100]
Out[62]= 0.28407904384041229602829183239312616909108808844573758275916266615`
          504587735148455373037841775223162587
```

Using `Recognize`, we readily find and check that $\mathcal{R}(e^{-2\pi}) = -1/2 - 5^{1/2}/2 + ((5 + 5^{1/2})/2)^{1/2}$.

```
In[63]:= Recognize[r, 12, x]
Out[63]= 1 - 2 x - 6 x2 + 2 x3 + x4
In[64]:= x /. Solve[% == 0, x][[-1]]
Out[64]=  $\frac{1}{2} \left( -1 - \sqrt{5} + \sqrt{2(5 + \sqrt{5})} \right)$ 
In[65]:= R[N[e-2π, 400], 400] - %
Out[65]= 0. × 10-401
```

It is again straightforward to automate the search for an algebraic number. The function `findAlgebraicValue` does this. We first find a polynomial of maximal degree o and then check the resulting root numerically to twice as many digits.

```
In[66]:= SetOptions[Roots, Cubics → False, Quartics → False];

findAlgebraicValue[a_, prec_] :=
  Block[{$MaxExtraPrecision = 1000, o = 2, polyα, α, β},
    α = R[N[e-aπ, prec], prec]; While[polyα = Recognize[α, o, t];
    β = Select[t /. Union[{ToRules[Roots[polyα == 0, t]]}], Abs[# - α] == 0 &];
    If[β === {}, True, β[[1]] - R[N[e-aπ, 2 prec], 2 prec] ≠ 0], o = o + 1]; β[[1]]
```

Next, we calculate the algebraic value of $\mathcal{R}(e^{-\pi})$ [11].

```
In[68]:= findAlgebraicValue[1, 200]
Out[68]= Root[1 - 14 #1 + 22 #12 - 22 #13 + 30 #14 + 22 #15 + 22 #16 + 14 #17 + #18 &, 4]
In[69]:= R[N[e-π, 200], 200] - %
Out[69]= -0. × 10-201
```

Unfortunately, *Mathematica* cannot find a radical expression of the root although it exists. No advanced Galois techniques are currently available.

```
In[70]:= ToRadicals[%]
```

```
Out[70]= Root[1 - 14 #1 + 22 #1^2 - 22 #1^3 + 30 #1^4 + 22 #1^5 + 22 #1^6 + 14 #1^7 + #1^8 &, 4]
```

```
In[71]:= RootReduce[ $\left(\frac{1}{8} (3 + \sqrt{5}) (\sqrt[4]{5} - 1) \left(\sqrt{10 + 2\sqrt{5}} - (3 + \sqrt[4]{5}) (\sqrt[4]{5} - 1)\right)\right)$ ]
```

```
Out[71]= Root[1 - 14 #1 + 22 #1^2 - 22 #1^3 + 30 #1^4 + 22 #1^5 + 22 #1^6 + 14 #1^7 + #1^8 &, 4]
```

Again using the previous modular equations, we can easily calculate (without time-consuming calls to *Recognize*) exact values for $\mathcal{R}(e^{-k\pi})$, $k \in \mathbb{N}$. Next, we calculate the first 10 values.

```
In[72]:= tab = Table[vSols = v /. Solve[findModularEquation[μ][1] == 0, v];
      δs = (vSols /. u → R[N[e-π, 200], 200]) - R[N[e-μ π, 200], 200];
      R[e-μ π] = (vSols[[Position[δs, _? (# == 0 &)][1, 1]]) /.
      u → Root[1 - 14 # + 22 #^2 - 22 #^3 + 30 #^4 + 22 #^5 + 22 #^6 + 14 #^7 + #^8 &,
      4]), {μ, 10};
```

These are the degrees of the resulting root-objects (this is sequence A082682 of the On-Line Encyclopedia of Integer Sequences).

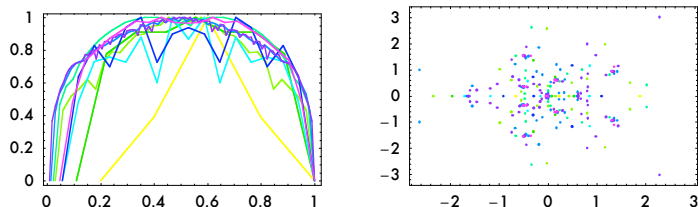
```
In[73]:= Exponent[#2, 1][z], z] & /@ tab
```

```
Out[73]= {8, 4, 32, 8, 40, 16, 64, 16, 96, 20}
```

On a logarithmic scale, the coefficients of the polynomials seem to approach a limit distribution. The right graphic shows the zeros of the defining polynomials in the complex plane.

```
In[74]:= Show[GraphicsArray[{Graphics[Table[{Hue[k/12],
      Line[Function[c1, MapIndexed[ $\left\{\frac{\#2[[1]]}{\text{Length}[c1]}, \frac{\#1}{\text{Max}[c1]}\right\}$  &, c1]] [
      Log[Abs[CoefficientList[tab[[k]][2, 1][z], z]]]], {k, 10}],
      PlotRange → All, Frame → True], Graphics[{PointSize[0.01],
      MapIndexed[ $\left\{\text{Hue}\left[\frac{\#2[[1]]}{12}\right], (\text{Point}[\{\text{Re}[\#1], \text{Im}[\#1]\}]) \& \right\}$  /@
      N[z /. Solve[#2, 1][z] == z, z]] &, tab]],
      PlotRange → Automatic, Frame → True]]]
```

From In[74]:=



This concludes our discussion on new modular identities; however, we will come back to the Rogers–Ramanujan continued fraction at a later date.

■ References

- [1] D. H. Bailey, "Finding New Mathematical Identities via Numerical Computations," *ACM SIGNUM*, **33**, 1998 pp. 17–22.
- [2] D. H. Bailey and J. M. Borwein in B. Engquist and W. Schmid, eds., *Mathematics Unlimited—2001 and Beyond*, Berlin: Springer-Verlag, 2001.
- [3] J. M. Borwein and R. M. Corless, "Emerging Tools for Experimental Mathematics," *American Mathematical Monthly*, **106**, 1999 pp. 889–909.
- [4] L. J. Rogers, "Second Memoir on the Expansion of Certain Infinite Products," *Proceedings of the London Mathematical Society*, **25**, 1894 pp. 318–343.
- [5] B. C. Berndt, *Ramanujan's Notebooks*, Vol. 5, New York: Springer-Verlag, 1998.
- [6] G. E. Andrews, B. C. Berndt, L. Jacobsen, and R. L. Lamphere, "The Continued Fractions Found in the Unorganized Portion of Ramanujan's Notebooks," *Memoirs of the American Mathematical Society*, **99**, 1992.
- [7] J. McLaughlin and D. Bowman, "On the Divergence of the Rogers-Ramanujan Continued Fraction on the Unit Circle." (October 15, 2001), arxiv.org/PS_cache/math/pdf/0107/0107043.pdf.
- [8] J. Yi, "Evaluations of the Rogers-Ramanujan Continued Fraction $R(q)$ by Modular Equations," *Acta Arithmetica*, **97**, 2001a pp. 103–127.
- [9] B. C. Berndt and H. H. Chan, "Some Values for the Rogers-Ramanujan Continued Fraction," *Canadian Journal of Mathematics*, **47**, 1995 pp. 897–914.
- [10] B. C. Berndt, H. H. Chan, S.-S. Huang, S.-Y. Kang, J. Sohn, and S. H. Son, "The Rogers-Ramanujan Continued Fraction," *Journal of Computational and Applied Mathematics*, **105**, 1999 pp. 9–24.
- [11] J. Yi, "Modular Equations for the Rogers-Ramanujan Continued Fraction and the Dedekind Eta-Function," *Ramanujan Journal*, **5**, 2001b pp. 377–384.

Michael Trott

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

Learning about Differential Equations from Their Symmetries

Scott A. Herod

Symmetries form the basis of the packages `DSolve` and `PDSolve1`. A knowledge of symmetries facilitates the understanding and analysis of solutions of differential equations and an understanding of the techniques used to solve the equations. This article introduces a package, `MathSym`, that assists in the computation of symmetries. `MathSym`'s flexibility and utility are illustrated through four examples, two from ordinary differential equations and two from partial differential equations.

■ Introduction

Generally, we teach our sophomore engineering students a course in techniques for solving differential equations. It often appears to the students that we are offering a disjoint collection of methods that we expect them to apply to a sequence of contrived examples. Later in their education many of these students are introduced to partial differential equations and again they are besieged by a multitude of techniques and “special” solutions. Almost all of the methods that we teach can be derived from one basic idea: the existence of symmetries of differential equations.

In the late 1870s Sophus Lie, working at the University of Christiania (now Oslo), applied his theories of *transformation groups* to differential equations. It was well known at the time that Abel's theory on the roots of polynomials was best understood in terms of the theories of Galois regarding symmetries of polynomial equations. Lie believed that his theories of continuous transformation groups would lead to a similar description of solutions to differential equations. He soon realized that many of the standard techniques for solving differential equations could be subordinated to a general method. In [1] Lie wrote that “the foundation of this method is the concept of an infinitesimal transformation and closely related to it the concept of a one-parameter group” (translation by F. Schwartz in [2]).

The transformations that Lie was writing about are usually called *symmetries*. They are found by posing the question: “What are the transformations of the

variables in a differential equation that map solutions of the equation to other solutions of the equation?” This is the fundamental question that we should always keep in mind when discussing the symmetries of a differential equation.

A symmetry of an ordinary differential equation can be used to reduce the order of the equation. My first example shows how this may be done. Also, Alexei Bocharov discusses *Mathematica*’s implementation of this technique in [3]. For partial differential equations, I will show how knowledge of a symmetry can be used to reduce the number of independent variables by one. So, for example, an equation in two independent variables can be converted to an ordinary differential equation.

Answering the question of what are the possible symmetries of any given differential equation leads to a massive computational problem. For this reason Lie’s technique was rarely used until recently. The advent of powerful computer algebra systems has made it feasible to carry out such long calculations. *MathSym* is a *Mathematica* package that performs many of the computations necessary to apply Lie’s technique. In addition, it incorporates ideas from Gröbner bases to reduce the equations, which must be solved in order to compute the symmetries of an equation.

I will not give a full description of the method of symmetry reduction of differential equations. Rather, I suggest the following references for someone interested in learning more about these techniques. The mathematical details of Lie’s technique may be found in [4, 5, 6, 7]. A description of the use of symmetries in *Mathematica* appears in [3]. For a discussion of Gröbner bases, see [8] and [9]. Application of symmetry reduction to two equations from fluid mechanics may be found in [10].

■ Reductions of Differential Equations: Scaling Symmetries

As an example of Lie’s technique, consider the standard linear second-order ordinary differential equation that we discuss in sophomore differential equations,

$$y''(x) + a y'(x) + b y(x) = 0. \quad (1)$$

If we rescale y by an arbitrary constant α , making the change of variable $\tilde{y} = \alpha y$, we are left with exactly the same equation,

$$\alpha \tilde{y}''(x) + \alpha a \tilde{y}'(x) + \alpha b \tilde{y}(x) = 0.$$

We say in this case that equation (1) is *scale invariant* and that we have identified a *scaling symmetry* of the equation.

In order to apply this scale symmetry to help us solve equation (1), we rewrite the equation as a pair of first-order equations,

$$u'(x) = v(x) \quad (2)$$

$$v'(x) = -a v(x) - b u(x), \quad (3)$$

and then ask: “Are there any quantities that are left invariant by the scaling symmetry?” Computation of such an invariant is algorithmic, but we will merely notice that $w = \tilde{v} / \tilde{u} = (\alpha v) / (\alpha u)$ is identical in both the original and in the new coordinate system. Assuming that w is a function of x , we next write the differential equations that $w(x)$ must satisfy. We do this by replacing v with the product uw in equations (2) and (3) to get

$$u'(x) = v(x) = w(x) u(x) \quad (4)$$

and

$$v'(x) = u'(x) w(x) + w'(x) u(x) = u(x) w(x)^2 + w'(x) u(x) = -a u(x) w(x) - b u(x).$$

This implies that

$$w'(x) = -b - a w(x) - w(x)^2. \quad (5)$$

We have converted the original problem to two integrals, namely

$$\int \frac{dw}{b + a w + w^2} = - \int dx$$

and, once $w(x)$ is known,

$$\int \frac{du}{u} = \int w(x) dx.$$

However, we recognize that we can actually say a lot about equation (5). Namely, it is the Riccati equation that corresponds to equation (1). Equation (4) is the Riccati transformation. Furthermore, if we choose the two special solutions to equation (5) given by the roots of the right-hand side and then solve equation (4) for $u(x)$, we have recovered the standard technique for solving linear constant coefficient differential equations that is taught in the sophomore course.

Our second example is the heat equation in one spatial dimension,

$$u_t = u_{xx}. \quad (6)$$

It is preserved by the change of variables given by the scalings $\tilde{u} = \beta u$, $\tilde{x} = \alpha x$, and $\tilde{t} = \alpha^2 t$. If we again search for quantities that are invariant under these scalings, we discover the new variables $s = -x^2 / (4t)$ and $w = u / t^{\gamma/2}$ where γ satisfies $\beta = \alpha^\gamma$. We assume that w is a function of s and compute the equation for w that arises by insisting that u be a solution of the heat equation. Setting $u(x, t) = t^{\gamma/2} w(s)$ and substituting into equation (6) yields an ordinary differential equation for w ,

$$s w''(s) + \left(\frac{1}{2} - s \right) w'(s) + \frac{\gamma}{2} w(s) = 0. \quad (7)$$

This equation is Kummer's equation and the solution of it may be expressed in terms of confluent hypergeometric functions, $M(-\frac{\gamma}{2}, \frac{1}{2}, s)$ and $U(-\frac{\gamma}{2}, \frac{1}{2}, s)$. *Mathematica* knows these two functions as `Hypergeometric1F1` $[-\frac{\gamma}{2}, \frac{1}{2}, s]$ and `HypergeometricU` $[-\frac{\gamma}{2}, \frac{1}{2}, s]$. We can use solutions to Kummer's equation to plot scale invariant solutions of the heat equation. Figure 1 is the result of setting $\gamma = -12.1$. It was generated with the following routine.

```

In[1]:= dl[g_] := DensityPlot[
  t^(g/2) Hypergeometric1F1[-g/2, 1/2, (-x^2)/(4 t)], {x, -3, 3},
  {t, 0.001, 0.6}, PlotPoints -> 250, FrameLabel -> {"x", "t", "", ""},
  ColorFunction -> (GrayLevel[#] &), Mesh -> False];
dl[-12.1]

```

From In[1]:=

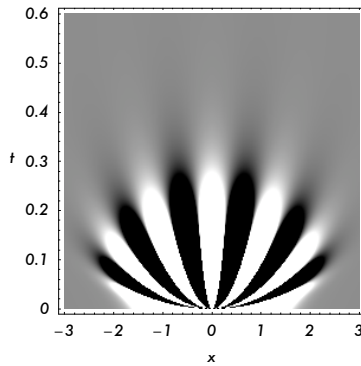


Figure 1. A scale invariant solution of the heat equation.

The horizontal axis gives the space variable x and the vertical axis is t . White sections are high temperature regions while dark regions are cooler.

■ Application of MathSym to Analyzing an Ordinary Differential Equation

In the previous section we used a scaling symmetry to help understand the solutions of a pair of differential equations. In each case, the scaling symmetry was found by inspection. Here I present the computation of the complete set of *point symmetries* for two additional differential equations. Our third example is a nonlinear ordinary differential that we analyze using its two symmetries. The final example is the partial differential equation known as the cubic nonlinear Schrödinger equation [11].

Example three is the ordinary differential equation

$$w''(x) + a w(x) w'(x) + b w(x)^3 = 0 \quad (8)$$

that arises in the study of nonlinear water wave equations. I also show that we can use its two symmetries to begin to learn something about the structure of its solutions.

MathSym returns a system of equations, the *determining equations*, whose solutions generate the symmetries of equation (8). Internally, the MathSym package denotes all independent variables in an equation as x_i and dependent variables as u_i . This way it can be run on systems of equations with arbitrary numbers of independent and dependent variables without needing to know how to treat

different variable names. Furthermore, constants are represented as $s[i]$ internally and printed as $s(i)$. With this notation, constants are treated correctly by *Mathematica*'s differentiation routine *Dt*. *MathSym*'s output is the following list of determining equations.

$$\begin{aligned} -\frac{\partial^2 \xi_1}{\partial u_1^2} &= 0 \\ 2 u_1 s(1) \frac{\partial \xi_1}{\partial u_1} + \frac{\partial^2 \eta_1}{\partial u_1^2} - 2 \frac{\partial^2 \xi_1}{\partial x_1} \partial u_1 &= 0 \\ 3 u_1^2 \eta_1 s(2) - u_1^3 s(2) \frac{\partial \eta_1}{\partial u_1} + u_1 s(1) \frac{\partial \eta_1}{\partial x_1} + 2 u_1^3 s(2) \frac{\partial \xi_1}{\partial x_1} + \frac{\partial^2 \eta_1}{\partial x_1^2} &= 0 \\ \eta_1 s(1) + 3 u_1^3 s(2) \frac{\partial \xi_1}{\partial u_1} + u_1 s(1) \frac{\partial \xi_1}{\partial x_1} + 2 \frac{\partial^2 \eta_1}{\partial x_1} \partial u_1 - \frac{\partial^2 \xi_1}{\partial x_1^2} &= 0 \end{aligned}$$

With the output from *MathSym* we can continue our analysis of equation (8). First, we solve the determining equations:

$$\xi_1 = c_1 + c_2 x_1 \quad (9)$$

$$\eta_1 = -c_2 u_1. \quad (10)$$

The functions ξ_1 and η_1 determine two symmetries that can be used to convert equation (8) into two integrals. The reader is directed to similar computations for the Blasius boundary layer equation which appear on pages 118–120 of [4].

We begin by considering the symmetry that occurs because of the c_1 term. Setting $\xi_1 = c_1$ and $\eta_1 = 0$ produces a transformation $\tilde{x} = x + c_1 \varepsilon$ and $\tilde{w} = w$. We next look for two quantities that do not change under this transformation. Obvious choices are $u = w$ and $v = w'$. If we assume that v is a function of u and write the differential equation for $v(u)$ that arises by insisting that $w(x)$ satisfy equation (8), we find

$$v(u) v'(u) + a u v(u) + b u^3 = 0. \quad (11)$$

This is a standard reduction of order for autonomous equations that may be found in a sophomore differential equations text such as [12].

This equation in u and v has a symmetry that is generated by the constant c_2 appearing in equations (9) and (10). From this symmetry we can derive new variables $r = v/u^2$ and $s = \ln v$ and consider s as a function of r . In terms of r and $s(r)$, equation (11) becomes

$$\frac{ds}{dr} = \frac{ar + b}{2r^3 + ar^2 + br}. \quad (12)$$

We have now converted the problem of solving the original equation into two integrations. First we find s as a function of r giving us a solution of equation (12) and hence of equation (11). Then we return to the original variables and have $w'(x)$ implicitly as a function of $w(x)$. Integrating again gives a relationship between w and x .

We can make *Mathematica* carry out some of these computations. First we will ask that it determine a solution to equation (12) by integrating both sides of the equation.

```
In[3]:= soln = (Integrate[s'[r], r] ==  
             Integrate[(a r + b) / (2 r^3 + a r^2 + b r), r] + C[1])
```

$$\text{Out[3]} = s[r] = \frac{a \operatorname{ArcTan}\left[\frac{a+4r}{\sqrt{-a^2+8b}}\right]}{\sqrt{-a^2+8b}} + C[1] + \operatorname{Log}[r] - \frac{1}{2} \operatorname{Log}[b + ar + 2r^2]$$

In the equation for $s(r)$ we can return to the original variables $w(x)$ and $w'(x)$.

```
In[4]:= soln /. {s[r] -> Log[w'[x]], r -> w[x] / w[x]^2}
```

$$\text{Out[4]} = \operatorname{Log}[w'[x]] = \frac{a \operatorname{ArcTan}\left[\frac{a + \frac{4w'[x]}{w[x]^2}}{\sqrt{-a^2+8b}}\right]}{\sqrt{-a^2+8b}} + C[1] + \operatorname{Log}\left[\frac{w'[x]}{w[x]^2}\right] - \frac{1}{2} \operatorname{Log}\left[b + \frac{aw'[x]}{w[x]^2} + \frac{2w'[x]^2}{w[x]^4}\right]$$

What results is an implicit relationship between w and w' , and while *MathSym* has been successful in generating the symmetries of equation (8), it still is a challenge to solve this equation.

■ Application of MathSym to Analyzing a Partial Differential Equation

As a final example, I will discuss the computation of the symmetries of the cubic nonlinear Schrödinger equation, a complex valued partial differential equation. Using the symmetries of the equation it is possible to generate exact solutions by a couple of different methods. First since symmetries of an equation transform its solutions to other solutions, I will demonstrate a family of solutions that are the transforms of a spatially invariant solution. Also, I will show that the symmetries can be used to reduce the partial differential equation to an ordinary differential equation in a couple of different ways.

For ordinary differential equations, it is theoretically possible to reduce the order of the equation by one for every symmetry of the equation. So, a second-order equation can be reduced to two integrals if two symmetries are known. Of course, as we saw above, in practice it can be difficult to perform the necessary computations. For a partial differential equation, it is generally not possible to get the full solution set from just the knowledge of several symmetries. However, by looking for fixed points of a given symmetry, we can find reductions of the equation and often special solutions.

The cubic nonlinear Schrödinger equation is derived in descriptions of nonlinear optics, water waves, and plasma physics [11]. Usually, the equation is written in terms of a complex valued function $\Psi(x, t)$. The equation is

$$i\Psi_t + \Psi_{xx} + 2|\Psi|^2\Psi = 0.$$

When computing the symmetries, we set $\Psi(x, t) = u(x, t) + i v(x, t)$ and use the system of equations

$$\begin{aligned} -v_t + u_{xx} + 2(u^2 + v^2)u &= 0 \\ u_t + v_{xx} + 2(u^2 + v^2)v &= 0 \end{aligned}$$

which we denote as NLS.

Again, `MathSym` returns a system of linear coupled partial differential equations for the generators of the symmetries. `MathSym` uses the convention that the independent variables x and t are stored and printed as x_1 and x_2 . The dependent variables u and v are represented by u_1 and u_2 . The generators of the symmetries are ξ_1 , ξ_2 , η_1 , and η_2 .

$$\begin{aligned} -(\eta_2 u_1) + \eta_1 u_2 - u_1^2 \frac{\partial \eta_1}{\partial u_2} - u_2^2 \frac{\partial \eta_1}{\partial u_2} &= 0 \\ \eta_1 u_1 + \eta_2 u_2 - u_1^2 \frac{\partial \eta_2}{\partial u_2} - u_2^2 \frac{\partial \eta_2}{\partial u_2} &= 0 \\ \frac{\partial \xi_1}{\partial u_2} &= 0 \\ \frac{\partial \xi_2}{\partial u_2} &= 0 \\ -(\eta_1 u_1) - \eta_2 u_2 + u_1^2 \frac{\partial \eta_1}{\partial u_1} + u_2^2 \frac{\partial \eta_1}{\partial u_1} &= 0 \\ \eta_2 u_1 - \eta_1 u_2 - u_1^2 \frac{\partial \eta_2}{\partial u_1} - u_2^2 \frac{\partial \eta_2}{\partial u_1} &= 0 \\ \frac{\partial \xi_1}{\partial u_1} &= 0 \\ \frac{\partial \xi_2}{\partial u_1} &= 0 \\ \frac{\partial \eta_1}{\partial x_2} &= 0 \\ \frac{\partial \eta_2}{\partial x_2} &= 0 \\ 2 \eta_1 u_1 + 2 \eta_2 u_2 + u_1^2 \frac{\partial \xi_2}{\partial x_2} + u_2^2 \frac{\partial \xi_2}{\partial x_2} &= 0 \\ u_2 \frac{\partial \xi_1}{\partial x_2} + 2 \frac{\partial \eta_1}{\partial x_1} &= 0 \\ u_1 \frac{\partial \eta_1}{\partial x_1} + u_2 \frac{\partial \eta_2}{\partial x_1} &= 0 \end{aligned}$$

$$\begin{aligned}\eta_1 u_1 + \eta_2 u_2 + u_1^2 \frac{\partial \xi_1}{\partial x_1} + u_2^2 \frac{\partial \xi_1}{\partial x_1} &= 0 \\ \frac{\partial \xi_2}{\partial x_1} &= 0 \\ \frac{\partial^2 \eta_1}{\partial x_1^2} &= 0\end{aligned}$$

MathSym has been successful in generating and reducing these equations. The solution of the determining equations is

$$\begin{aligned}\xi_1 &= c_0 + c_1 x + 2 c_2 t \\ \xi_2 &= c_3 + 2 c_1 t \\ \eta_1 &= c_4 v - c_1 u - c_2 x v \\ \eta_2 &= -c_4 u - c_1 v + c_2 x u\end{aligned}$$

where the c_i are arbitrary constants. For clarity we have returned to the original variables x , t , u , and v . From these generators we can compute the symmetries for the nonlinear Schrödinger equation and with the symmetries we can derive solutions to NLS using at least two different strategies.

Recall that symmetries map solutions to solutions so knowledge of a solution and a symmetry allows us to generate a family of new solutions. As an example, let us use the *Galilean boost*, which is represented above by the constant c_2 . We can compute a transformation by setting $c_2 = 1$ and all of the other constants in the list of generators for the symmetries to zero. The transformation corresponding to this choice of constants is given by the change to the new variables $\tilde{x}$, $\tilde{t}$, $\tilde{u}$, and $\tilde{v}$ where

$$\tilde{x} = x + 2 \varepsilon t \quad (13)$$

$$\tilde{t} = t \quad (14)$$

$$\tilde{u} = \sqrt{u^2 + v^2} \cos\left(\frac{1}{2} t \varepsilon^2 + x \varepsilon + \tan^{-1}(v/u)\right) \quad (15)$$

$$\tilde{v} = \sqrt{u^2 + v^2} \sin\left(\frac{1}{2} t \varepsilon^2 + x \varepsilon + \tan^{-1}(v/u)\right). \quad (16)$$

The point in this exercise was to generate transformations of the variables appearing in the nonlinear Schrödinger equation that sent solutions of the equation to other solutions. This means that any solution to NLS can be used to generate a family of solutions.

It is straightforward to check that a solution of NLS is

$$u(t) = k \cos(2 k^2 t) \quad (17)$$

$$v(t) = k \sin(2 k^2 t) \quad (18)$$

where k is an arbitrary constant. This solution is a prototypical nonlinear oscillator where the frequency is a function of the amplitude.

The transformation in equations (13) to (16) gives new solutions to NLS when applied to equations (17) and (18). If we carry out these substitutions, we find

$$\begin{aligned}\tilde{u}(\tilde{t}) &= k \cos\left(2 k^2 \tilde{t} + \varepsilon \tilde{x} - \frac{1}{2} \varepsilon^2 \tilde{t}\right) \\ \tilde{v}(\tilde{t}) &= k \sin\left(2 k^2 \tilde{t} + \varepsilon \tilde{x} - \frac{1}{2} \varepsilon^2 \tilde{t}\right)\end{aligned}$$

which can be shown to satisfy NLS for any choices of k and ε . Images of the real parts for these two solutions are generated in the following cells.

```
In[5]:= Plot3D[Cos[2 t],{x,-10,10},{t,0,10},PlotPoints->50,
  AxesLabel->{"x","t","u(x,t)"},Mesh->False,
  PlotRange->{-2,2}]
```

From In[5]:=

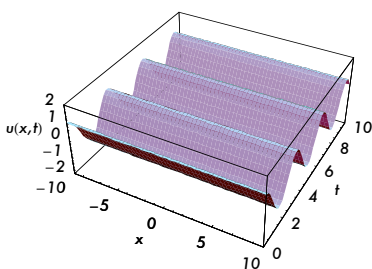


Figure 2. The real part of a spatially invariant solution of the nonlinear Schrödinger equation.

```
In[6]:= Plot3D[Cos[2 t + .3 x - (.3)^2/2 t],{x,-10,10},{t,0,10},
  PlotPoints->50,
  AxesLabel->{"x","t","u(x,t)"},Mesh->False,
  PlotRange->{-2,2}]
```

From In[6]:=

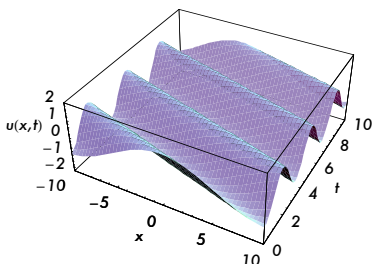


Figure 3. The real part of a transformed solution of the nonlinear Schrödinger equation.

Finally, we will use the symmetries to derive ordinary differential equations. In order to compute the reductions, we pose the question: “What are the solutions

of NLS that are invariant under a given symmetry?” Notice that this question is similar to the one that we asked to define the symmetries. A symmetry is a mapping of the solution set of an equation to itself. Reductions arise from looking at the invariant subset of that transformation.

To find these invariant solutions we look for the intersection of the solution sets of the original equation, here the NLS equation, and a pair of first-order, quasi-linear partial differential equations, which are

$$\begin{aligned}\xi_1 \frac{\partial u}{\partial x} + \xi_2 \frac{\partial u}{\partial t} &= \eta_1 \\ \xi_2 \frac{\partial v}{\partial x} + \xi_2 \frac{\partial v}{\partial t} &= \eta_2.\end{aligned}$$

These equations are known as the *invariant surface condition equations*. We will solve this new pair of equations using the method of characteristics and then substitute the solutions into the NLS equation.

We start with $\xi_1 = c_0$, $\xi_2 = c_3$, and $\eta_1 = \eta_2 = 0$. When we solve the invariant surface equations, we find that along characteristic curves, $s = x - c t$, u and v must be constant. Thus, $u(x, t) = f(s)$ and $v(x, t) = g(s)$ for arbitrary functions f and g . Here $c = c_0 / c_3$. The functions f and g must be determined by the NLS equation. You perhaps recognize f and g as being the functions that one must find in order to compute the traveling wave solutions of the NLS equation.

Substituting $u(x, t) = f(s)$ and $v(x, t) = g(s)$ into the NLS equation gives us two equations that we must solve in order to find f and g . These two new equations are now ordinary differential equations rather than the original partial differential equations. They are

$$\begin{aligned}c g' + f'' + 2(f^2 + g^2) f &= 0 \\ -c f' + g'' + 2(f^2 + g^2) g &= 0.\end{aligned}$$

We can approximate solutions to this pair of ordinary differential equations by changing to polar coordinates $r(s)^2 = f(s)^2 + g(s)^2$ and $\theta(s) = \tan^{-1}(g(s)/f(s))$ and integrating twice. The new equations are

$$\begin{aligned}\frac{1}{2} (r')^2 + \frac{1}{2} \left[r^4 + \left(\frac{c}{2} \right)^2 r^2 + \left(\frac{k}{r} \right)^2 \right] &= T \\ \theta' &= \frac{c}{2} + \frac{k}{r^2}\end{aligned}$$

with k and T constants of the integration. The form of the equation for $r(s)$ was chosen on purpose to remind us of the equation from classical mechanics,

$$\frac{1}{2} m v^2 + \text{P.E.} = \text{Energy.} \quad (19)$$

If we let $\xi_1 = c_1 x$, $\xi_2 = 2 c_1 t$, $\eta_1 = -c_1 u$, and $\eta_2 = -c_1 v$, we find that, along characteristic curves $s = x^2 / t$, invariant solutions of the NLS equation multiplied by x must be constant. That is, $x u(x, t) = f(s)$ and $x v(x, t) = g(s)$. Note that these are different f , g , and s than those that we just used. Substituting into the

NLS equation gives a pair of coupled ordinary differential equations that f and g must solve. This time these equations are

$$\begin{aligned} s^2 g' + 4 s^2 f'' + 2 f - 2 s f' + (f^2 + g^2) f &= 0 \\ -s^2 f' + 4 s^2 g'' + 2 g - 2 s g' + (f^2 + g^2) g &= 0. \end{aligned}$$

Our last reduction will come from letting $\xi_1 = 2t$, $\xi_2 = 0$, $\eta_1 = -xv$, and $\eta_2 = xu$. We will switch to a polar representation immediately here. The resulting pair of ordinary differential equations that we derive are

$$\begin{aligned} 2 s r'(s) + r(s) &= 0 \\ \left(\frac{\theta(s)}{s} \right)' + 8 r(s)^2 &= 0. \end{aligned}$$

We can solve these two equations to get

$$\begin{aligned} r(s) &= \frac{c_1}{\sqrt{s}} \\ \theta(s) &= -4 c_2 s - 8 c_1 s \ln s. \end{aligned}$$

In terms of the original variables this is

$$\begin{aligned} u(x, t) &= \frac{c_1}{\sqrt{t}} \cos\left(\frac{x^2}{4t} + c_2 + 2 c_1 \ln t\right) \\ v(x, t) &= \frac{c_1}{\sqrt{t}} \sin\left(\frac{x^2}{4t} + c_2 + 2 c_1 \ln t\right). \end{aligned}$$

Here it is nice to have *Mathematica* check our computations. First we will derive the equations. Notice that we are using a little i instead of the capital I that *Mathematica* usually uses. This is done so that we can separate using a `CoefficientList`.

```
In[7]:= SetAttributes[{i, x, t}, Constant];
eqn = (i Dt[psi, t] + Dt[psi, x, x] + 2 Abs[psi]^2 psi);
eqn = (eqn /. psi -> u[x, t] + i v[x, t]) // Factor;
eqn = eqn /. Abs[_] -> Sqrt[u[x, t]^2 + v[x, t]^2];
eqn = CoefficientList[(eqn /. i^2 -> -1), i]

Out[11]= {2 u[x, t] (u[x, t]^2 + v[x, t]^2) - v^(0,1)[x, t] + u^(2,0)[x, t],
          2 v[x, t] (u[x, t]^2 + v[x, t]^2) + u^(0,1)[x, t] + v^(2,0)[x, t]}
```

Before we substitute for u and v , we need to “undo” the differentiation that occurred. It would also be nice to strip the arguments x and t from u and v on the cubic term. We will do this using two functions `UnDt` and `UnArg`. Here is their syntax.

```
In[12]:= UnList[{x__}] := Sequence[x];
UnDt[f_] := f /. Derivative[order__][func_] [vars__] ->
Dt[func, UnList[Transpose[{List[vars], List[order]}]]];
UnArg[stuff_, func_] := stuff /. func[_] -> func;
UnArg[stuff_, {func_}] := UnArg[stuff, func];
UnArg[stuff_, {func_, morefuncs__}] :=
UnArg[UnArg[stuff, func], {morefuncs}];
```

We apply these two functions.

```
In[17]:= eqn = UnArg[UnDt[eqn], {u, v}]
Out[17]= {2 u (u^2 + v^2) + Dt[u, {x, 2}] - Dt[v, t],
          2 v (u^2 + v^2) + Dt[u, t] + Dt[v, {x, 2}]}
```

We can now substitute our solution in and run the result through Simplify.

```
In[18]:= eqn /. {u -> (c[1] / Sqrt[t]) Cos[c[2] + 2 c[1]^2 Log[t] + x^2 / (4 t)], v ->
              (c[1] / Sqrt[t]) Sin[c[2] + 2 c[1]^2 Log[t] + x^2 / (4 t)]} // Simplify
Out[18]= {0, 0}
```

So it works. Let us see what some pictures look like. The constant $c[2]$ is only a phase shift so we will set it to zero. $c[1]$ affects the amplitude and frequency and we will let it be one. Here is the command to generate a density plot of the real part of the solution that is given by $u(x, t)$.

```
In[19]:= DensityPlot[(1 / Sqrt[t]) Cos[2 Log[t] + x^2 / (4 t)],
                     {x, -10, 10}, {t, 0.01, 3.0}, PlotPoints -> 250, AxesLabel -> {"x", "t"},
                     FrameLabel -> {"x", "t", "", ""},
                     ColorFunction -> (GrayLevel[#] &), Mesh -> False]
```

From In[19]:=

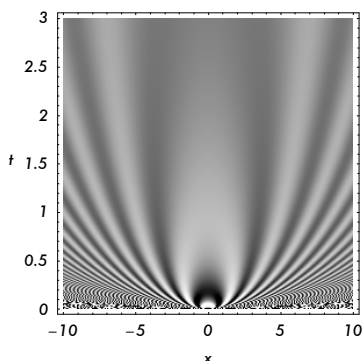


Figure 4. The real part of a boost invariant solution of the nonlinear Schrödinger equation.

■ Closing Comments

Symmetries of differential equations are useful in understanding their solutions and as a way to understand the techniques that we use to find the solutions. We have shown how some of these techniques can be derived from the symmetries of the equations and how generalizations of the methods for solving ordinary differential equations lead to exact solutions for a couple of partial differential equations.

To assist in determining the symmetries for equations, we have also introduced a package `MathSym` that performs many of the calculations. In addition to the

calculation of the Lie symmetries for differential equations that we have demonstrated here, `MathSym` can use ideas from differential Gröbner bases to generate a reduced form of the determining equations. For a discussion of algorithms similar to the ones that we have used, see [9].

`MathSym` can also generate the determining equations for the conditional symmetries first introduced by Bluman and Cole in 1969 [13]. Reductions with respect to conditional symmetries include the reductions derived using the *direct method* of Clarkson and Kruskal [14].

Finally, I do not consider `MathSym` a finished product. There are generalizations of Lie's technique that the package does not currently address. A lot of recent work in symmetry techniques for differential equations focuses on *generalized* symmetries and symmetries of difference equations. It is my intention to incorporate some of these ideas into future versions of the `MathSym` package.

■ Input Files

`MathSym` uses input files for the information that it needs. I decided to do this because the amount of information that the package needs is reasonably large, and I have found it useful to keep a record of all of the necessary information needed to run a particular problem.

The first two input files do not include the three optional arguments that occur in the input file for the nonlinear Schrödinger equation. The defaults for these variables are discussed in the documentation, which is part of the standard distribution of `MathSym`. The first listing is for the example of the heat equation discussed in the first section, Reductions of Differential Equations: Scaling Symmetries. It is also included in Additional Material in the file named `Heat_eqn.m`. The second listing contains the information for the ordinary differential equation described in Application of `MathSym` to Analyzing an Ordinary Differential Equation, and is contained in the file `ODE_eqn.m`. Finally, the input file for the nonlinear Schrödinger equation from the section Application of `MathSym` to Analyzing a Partial Differential Equation is listed in the third input cell. This information is also contained in the file `NLS_eqn.m` in Additional Material.

```
In[20]:= eqname = "the 1x1 Dimensional Heat Equation equation";
         numeqn = 1;
         numind = 2;
         numdep = 1;
         numpar = 0;
         diffeqn = {Dt[u1, x[2]] - Dt[u1, {x[1], 2}]};
         removedvar = {Dt[u1, {x[1], 2}]};
```

```

In[27]:= eqname = "a nonlinear ODE ";
numeqn = 1;
numind = 1;
numdep = 1;
numpar = 0;
diffeqn = {Dt[u1, x[1], x[1]] + s[1] u1 Dt[u1, x[1]] + s[2] u1^3};
removedvar = {Dt[u1, x[1], x[1]]};

In[34]:= eqname = "the Nonlinear Schroedinger equation";
tech = "Classical";
splitlist = True;
sortstyle = "Lexicographical";
numeqn = 2;
numind = 2;
numdep = 2;
numpar = 0;
diffeqn = {Dt[u1, {x[1], 2}] - Dt[u2, x[2]] + 2 (u1 * u1 + u2 * u2) * u1,
           Dt[u2, {x[1], 2}] + Dt[u1, x[2]] + 2 (u1 * u1 + u2 * u2) * u2};
removedvar = {Dt[u1, {x[1], 2}], Dt[u2, {x[1], 2}]};

```

■ Acknowledgments

I wish to thank Willy Hereman and Elizabeth Mansfield for discussions of computer techniques used when computing symmetries and Mark J. Ablowitz and Peter Clarkson for comments on the use of symmetries of differential equations. Thanks to James H. Curry for reading and commenting on numerous versions of this manuscript. Finally, I wish to thank the staff at Wolfram Research, especially Alexei Bocharov, for help with the intricacies of *Mathematica*.

Supported in part by the Department of Energy under grant DOE DE-FG03-94ER25194.

■ References

- [1] S. Lie, "Über die Integration durch bestimmte Integrale von einer Klasse linearer partieller Differentialgleichungen," *Arch. for Math.*, **VI**, Heft 3, 1881 S. 328–368.
- [2] F. Schwartz, "Symmetries of Differential Equations: From Sophus Lie to Computer Algebra," *Siam Review*, **30**, 1988 pp. 450–481.
- [3] A. V. Bocharov, "Symbolic Solvers for Nonlinear Differential Equations," *The Mathematica Journal*, **3**(2), 1993 pp. 63–69.
- [4] G. W. Bluman and S. Kumei, *Symmetries and Differential Equations*, New York: Springer-Verlag, 1989.
- [5] P. J. Olver, *Applications of Lie Groups to Differential Equations*, New York: Springer-Verlag, 1986.
- [6] L. V. Ovsiannikov, *Group Analysis of Differential Equations* (W. F. Ames, ed.), New York: Academic Press, 1982.

- [7] N. H. Ibragimov, *Lie Group Analysis of Differential Equations*, Boca Raton: CRC Press, 1994.
- [8] G. Helzer, "Grobner Bases," *The Mathematica Journal*, **5**(1), 1995 pp. 61–73.
- [9] F. Schwartz, "An Algorithm for Determining the Size of Symmetry Groups," *Computing*, **49**, 1992 pp. 95–115.
- [10] S. A. Herod, "Families of Exact Solutions for the Barotropic Vorticity Equation," *PAM Technical Report*, 1995.
- [11] M. J. Ablowitz and P. A. Clarkson, *Solitons, Nonlinear Evolution Equations, and Inverse Scattering*, Cambridge: Cambridge University Press, 1991.
- [12] A. L. Rabenstein, *Elementary Differential Equations with Linear Algebra*, New York: Academic Press, 1975.
- [13] G. W. Bluman and J. D. Cole, "The General Solution of the Heat Equation," *Journal of Mathematical Mechanics*, **18**(11), 1969 pp. 1025–1042.
- [14] P. A. Clarkson and M. D. Kruskal, "New Similarity Solutions of the Boussinesq Equation," *Journal of Mathematical Physics*, **30**, 1989 pp. 2201–2213.

■ Additional Material

MathSym11.m
Heat_eqn.m
ODE_eqn.m
NLS_eqn.m
Heat_run.nb
NLS_run.nb
README
Documentation.nb

Available at www.mathematica-journal.com.

About the Author

Scott A. Herod did postdoctoral research at the University of Colorado in Boulder where his interests included differential equations, dynamical systems, and computational mathematics. He now works as a software consultant specializing in the video industry.

Scott A. Herod

Department of Applied Mathematics
University of Colorado, Boulder
Campus Box 526
Boulder, Colorado 80309-0526
scott.herod@colorado.edu

p-adic Arithmetic

Stany De Smedt

The *p*-adic numbers were introduced by K. Hensel in 1908 in his book *Theorie der algebraischen Zahlen*, Leipzig, 1908. In this article we present a package that does *p*-adic calculations using *Mathematica*. It allows addition, subtraction, multiplication, and division of *p*-adic numbers written in Hensel expansion. Functions, such as sqrt, log, exp, sin, cos, sinh, and cosh, are defined using their power series expansion. We also calculate the coefficients of a continuous function with respect to the Mahler and Vanderput bases, and implement the *p*-adic Newton method.

■ Introduction

Let *p* be a prime number, fixed once and for all. If *x* is any rational number other than 0, we can write *x* in the form $x = p^n \frac{a}{b}$, where *a*, *b* ∈ $\mathbb{Z}$ are relatively prime to *p* and *n* ∈ $\mathbb{Z}$. We now define $|x|_p = p^{-n}$ and $|0|_p = 0$, and $\text{ord}_p(x) = n$ and $\text{ord}_p(0) = +\infty$. They satisfy the following properties.

1. $|x|_p \geq 0$, $|x|_p = 0$ if and only if $x = 0$.
2. $|x + y|_p \leq \max\{|x|_p, |y|_p\}$ (the strong triangle inequality) with $|x + y|_p = \max\{|x|_p, |y|_p\}$ if $|x|_p \neq |y|_p$ (the isosceles triangle principle).
3. $|x \cdot y|_p = |x|_p \cdot |y|_p$.

With ord_p instead of $|\cdot|_p$, we get for 2 and 3:

$$\begin{aligned}\text{ord}_p(x + y) &\geq \min\{\text{ord}_p(x), \text{ord}_p(y)\} \\ \text{ord}_p(x \cdot y) &= \text{ord}_p(x) + \text{ord}_p(y)\end{aligned}$$

$|\cdot|_p$ is called the *p*-adic valuation; $\text{ord}_p(\cdot)$ is called the *p*-adic order.

Ostrowski proved that each nontrivial valuation on the field of rational numbers is equivalent either to the absolute value function or to some *p*-adic valuation.

Recall that two valuations $|\cdot|_1$ and $|\cdot|_2$ are equivalent if there exists a positive constant *c* such that $|\cdot|_2 = |\cdot|_1^c$. For a proof of this theorem, see [1].

The completion of the field $\mathbb{Q}$ of rational numbers with respect to the *p*-adic valuation $|\cdot|_p$ is called the field of *p*-adic numbers and will be denoted $\mathbb{Q}_p$. The set $\mathbb{Z}_p = \{x \in \mathbb{Q}_p \mid |x|_p \leq 1\}$ is the ring of *p*-adic integers.

It can be easily proved that each p -adic number x can be written in the form

$$x = \sum_{n=-f}^{\infty} a_n p^n,$$

where each a_n is one of the elements $0, 1, \dots, p-1$, and $f \in \mathbb{Z}$. This is called the *Hensel representation* of the p -adic numbers. Sometimes one uses, analogous to the ordinary decimal notation for real numbers, the notation

$$a_{-f} \dots a_{-3} a_{-2} a_{-1} a_0, a_1 a_2 \dots a_{n-1} a_n \dots$$

It is this representation, preceded with $< p >$ to denote the p -adic, that we will use in our package.

With this representation, one obtains for $x \in \mathbb{Q}_p$:

$$\text{ord}_p(x) = \begin{cases} +\infty & \text{if } a_i = 0 \text{ for all } i, \\ \min \{s \mid a_s \neq 0\} & \text{otherwise,} \end{cases}$$

and

$$|x|_p = p^{-\text{ord}_p(x)}.$$

The set of values of $|\cdot|_p$ is $\{0\} \cup \{p^n \mid n \in \mathbb{Z}\}$, and $\{p^n \mid n \in \mathbb{Z}\} = \{|x|_p \mid x \in \mathbb{Q}_p \setminus \{0\}\}$ is called the *valuation group* of $\mathbb{Q}_p$.

Every $x \in \mathbb{Z}_p$ can be written as $x = \sum_{n=0}^{\infty} a_n p^n$ with $a_n \in \{0, 1, \dots, p-1\}$. The elements with $a_n = 0$ for sufficiently large n can be identified with the nonnegative integers. Thus $\mathbb{N} \subset \mathbb{Z}_p$ and $\mathbb{Z}_p$ also contains $\mathbb{Z}$ as a subset. For example,

$$-1 = (p-1) + (p-1)p + (p-1)p^2 + \dots$$

Note also that every nonzero element x of $\mathbb{Q}_p$ can be written as $p^n y$, where $n \in \mathbb{Z}$ and $y \in \mathbb{Z}_p$ with $|y|_p = 1$. And just as with decimal fractions, a p -adic number $x = \sum_{i=N}^{\infty} a_i p^i$ is rational if and only if the sequence of the digits a_i is periodic from some index i on.

Due to the strong triangle inequality, we may use the following property, which is much easier than in the Archimedean case:

$$\sum_{i=1}^{\infty} a_i \text{ converges if and only if } \lim_{i \rightarrow \infty} a_i = 0.$$

In general we say that a sequence $a_1, \dots, a_n, \dots$ in $\mathbb{Q}_p$ converges to a if and only if $\lim_{n \rightarrow \infty} |a_n - a|_p = 0$ and we will write this as $\lim_{n \rightarrow \infty} a_n = a$.

For $\mathbb{Z}_p$ and $\mathbb{Q}_p$ we have the following topological properties, which we only present here for completeness.

1. $\mathbb{Z}_p$ is compact: each covering by means of open sets has a finite subcovering.

2. $\mathbb{Z}_p$ is complete: every Cauchy sequence converges.
3. $\mathbb{Z}$ is dense in $\mathbb{Z}_p$: each element of $\mathbb{Z}_p$ is the limit of elements of $\mathbb{Z}$.
4. $\mathbb{Q}_p$ is locally compact: each point has a compact neighborhood.
5. $\mathbb{Q}$ is dense in $\mathbb{Q}_p$.
6. $\mathbb{Q}_p$ is complete and separable: it has a countable dense subset.
7. $\mathbb{Q}_p$ is zero-dimensional: every neighborhood of a point contains an open and closed subset.
8. $\mathbb{Q}_p$ is totally disconnected: the only subsets that are connected as a metric space are the empty set and the singletons.

We now survey the different parts that are treated in this package. For more information on *p*-adic calculus, see [1] or [2].

■ *p*-adic Order and Valuation

This is the implementation of the previously defined functions ord_p and $|\cdot|_p$. `PadicOrder[s,p]` returns the number of times that *p* divides *s*, roughly speaking.

$$\text{PadicValue}[s, p] = 1 / p^{\text{PadicOrder}[s, p]}$$

Here are some examples.

```
In[1]:= Get["PadicArithmetic.m"];
In[2]:= PadicOrder[3/16, 2]
Out[2]= -4
In[3]:= PadicValue[144, 3]
Out[3]= 1/9
```

■ Conversion to *p*-adics

Next we calculate the Hensel expansion of a *p*-adic number to a given precision. One may think of `PadicN[n,p,k]` as being a bit like `N[n,k]`. The internal representation is given as `PadicRational[m,n,e,p]`. `PadicN[n,p]` gives an eight-digit *p*-adic number; `PadicN[n,p,k]` gives *k* digits. The numbers to the right of the radix point represent the positive (> 0) powers, and the numbers to the left represent the negative (≤ 0) powers.

The implementation to print these expansions is made in the third section of the package entitled "Printing of *p*-adics" where we use `PadicDigits[m,n,e,p]` as an auxiliary function to aid formatting.

Here are some examples.

```
In[4]:= PadicN[102356, 3, 12]
Out[4]= <3> 2 . 2 2 1 0 1 2 1 0 2 1 0

In[5]:= PadicN[1/6, 2, 10]
Out[5]= <2> 1 1 . 0 1 0 1 0 1 0 1

In[6]:= PadicN[17/1024, 2]
Out[6]= (<2>1 . 0 0 0 1 0 0 0)  $\frac{1}{2^{10}}$ 

In[7]:= PadicN[125048, 20, 6]
Out[7]= <20> 8 . 12 12 15 0 0
```

■ Basic p -adic Arithmetic

Addition and multiplication of p -adic numbers are defined in several cases whether or not both numbers are given in Hensel expansion. Note that addition and multiplication in $\mathbb{Q}_p$ can be defined by

$$\sum_{n=N}^{\infty} a_n p^n + \sum_{n=M}^{\infty} b_n p^n = \sum_{n=\min\{N,M\}}^{\infty} (a_n + b_n) p^n,$$

$$\left(\sum_{n=N}^{\infty} a_n p^n \right) \left(\sum_{n=M}^{\infty} b_n p^n \right) = \sum_{n=N+M}^{\infty} \left(\sum_{i=N}^{n-M} a_i b_{n-i} \right) p^n.$$

The left-hand expansions are Hensel expansions, but the right-hand ones, in general, are not.

Also p -adic exponentiation is implemented for the case of integer powers. We then define the p -adic version of some classical functions, such as log, exp, sin, cos, sinh, and cosh. This is done with the help of their power series expansions, which are completely analogous to the real case except that their regions of convergence might differ [1]. We have also defined square roots. The existence of square roots of a number x in $\mathbb{Q}_p$ depends on the following theorems.

Let $p \neq 2$. A p -adic number $x = \sum_{i=0}^{\infty} a_i p^i$ ($0 \leq a_i \leq p-1$, $a_0 \neq 0$) is a square if and only if a_0 is a square residue mod p .

Let $p = 2$. A 2-adic number x , $|x|_2 = 1$ is a square if and only if $x \equiv 1 \pmod{8}$.

Finally, we implement the p -adic Gamma function.

Here are some examples.

```
In[8]:= PadicN[10, 3, 5]
Out[8]= <3> 1 . 0 1 0 0
```

```
In[9]:= PadicN[20, 3, 8]
Out[9]= <3> 2 . 0 2 0 0 0 0 0
In[10]:= % + %%
Out[10]= <3> 0 . 1 0 1 0 0
In[11]:= PadicN[15, 5] + 17
Out[11]= <5> 2 . 1 1 0 0 0 0 0 0
In[12]:= PadicN[15, 5] * 17
Out[12]= <5> 0 . 1 0 2 0 0 0 0 0
In[13]:= PadicN[9, 2, 10] 2
Out[13]= <2> 0 . 1 0 0 1 0 0 0 0 0 0
In[14]:= PadicN[4, 3, 4]
Out[14]= <3> 1 . 1 0 0
In[15]:= Exp[%]
```

Exp::overflow : Number not in the region of convergence

```
In[16]:= Log[%%]
Out[16]= <3> 0 . 1 2 1 2 2
In[17]:= PadicN[15, 3, 4]
Out[17]= <3> 0 . 2 1 0 0
In[18]:= Exp[%]
Out[18]= <3> 1 . 2 1 0 1
In[19]:= PadicN[-1, 3, 4]
Out[19]= <3> 2 . 2 2 2
In[20]:= Sqrt[%]
```

Sqrt::nonexistence : Square root does not exist

```
In[21]:= PadicN[31, 5, 10]
Out[21]= <5> 1 . 1 1 0 0 0 0 0 0 0
In[22]:= Sqrt[%]
Out[22]= <5> 1 . 3 3 2 2 0 4 1 4 2 1
In[23]:= PadicGamma[15, 3]
Out[23]= -44844800
```

In[24]:= **PadicGamma**[-2, 3]

Out[24]= $\frac{1}{2}$

■ Booleans and Selectors on p -adic Rationals

Here we implement whether or not a given number is a p -adic number. The second example selects the different parts of a p -adic number. Next we implement the conversion of a p -adic number to a rational number. We also define the function **SqrtQ**[x, p] to test whether the square roots of x exist in $\mathbb{Q}_p$ or not.

Here are some examples.

In[25]:= **PadicN**[12, 3, 5] / **PadicN**[4, 3, 3]

Out[25]= <3> 0 . 1 0 0

In[26]:= **PadicToRational**[%]

Out[26]= 3

In[27]:= **PadicExponent**[%]

Out[27]= 1

In[28]:= **SqrtQ**[-1, 3]

Out[28]= False

■ Function Expansion

For the space $C(\mathbb{Z}_p \rightarrow \mathbb{Q}_p)$ of all continuous functions from $\mathbb{Z}_p$ to $\mathbb{Q}_p$ (we can simply transfer the definitions from classical analysis to non-Archimedean analysis by adapting the valuations), there exist two well-known bases. On the one hand, we have the *Mahler basis*, consisting of the polynomials $\binom{x}{n}$, $n \in \mathbb{N}$. On the other hand, there is the *Vanderput basis*, consisting of locally constant functions $e_n(x)$, where $e_0(x) = 1$ and for $n \neq 0$, e_n is the characteristic function of the open ball with center n and radius $\frac{1}{n}$. Each continuous function $f : \mathbb{Z}_p \rightarrow \mathbb{Q}_p$ can then be written as

$$f(x) = \sum_{n=0}^{\infty} a_n \binom{x}{n} \text{ with } a_n = \sum_{j=0}^n (-1)^{n-j} \binom{n}{j} f(j),$$

$$f(x) = \sum_{n=0}^{\infty} a_n e_n(x) \text{ with } a_0 = f(0) \text{ and } a_n = f(n) - f(n_-) \text{ for } n \neq 0.$$

Here n_- is defined as follows. For every $n \in \mathbb{N}_0$, there exists a p -adic expansion $n = a_0 + a_1 p + \cdots + a_N p^N$ with $a_N \neq 0$. Then $n_- = a_0 + a_1 p + \cdots + a_{N-1} p^{N-1}$.

In this section of the package we implement these expansions as `Mahler[f,x,n]` and `Vanderput[f,x,n,p]`, which calculate the first $n + 1$ coefficients of the function $f(x)$ according to the respective basis. Note that the Vanderput expansion depends on p , while the Mahler expansion does not.

Here are some examples.

`In[29]:= Mahler[x3 + x + 1, x, 5]`

0 : 1
1 : 2
2 : 6
3 : 6
4 : 0
5 : 0

`In[30]:= Vanderput[x3 + x + 1, x, 5, 3]`

0 : 1
1 : 2
2 : 10
3 : 30
4 : 66
5 : 120

`In[31]:= Vanderput[x3 + x + 1, x, 5, 5]`

0 : 1

1 : 2

2 : 10

3 : 30

4 : 68

5 : 130

■ VolkenbornIntegral

This is the p -adic equivalent of the classical integral. The *Volkenbornintegral* is defined as

$$\int_{\mathbb{Z}_p} f(x) dx = \lim_{n \rightarrow \infty} p^{-n} \sum_{k=0}^{p^n-1} f(k)$$

or equivalently

$$\int_{\mathbb{Z}_p} f(x) dx = \sum_{n=0}^{\infty} \frac{(-1)^n}{n+1} \Delta^n f(0).$$

Here Δ is the *forward difference quotient*: $\Delta f(x) = f(x+1) - f(x)$ and $\Delta^n f(x) = \Delta(\Delta^{n-1} f(x))$. We used this last expression for our implementation.

■ p -adic Newton Method

Here we implement the well-known *Newton method* to solve an algebraic equation $f(x) = 0$. The algorithm is the same as in the classical case, that is, we use the iterations

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}.$$

The main problem is to find an appropriate starting value since convergence might be rather slow. We tried to solve this based on the following theorem.

Let $F(x)$ be a polynomial with p -adic integer coefficients and let there exist $\gamma \in \mathbb{Z}_p$, such that $F(\gamma) \equiv 0 \pmod{p}$ and $F'(\gamma) \not\equiv 0 \pmod{p}$. Then there exists $\alpha \in \mathbb{Z}_p$, such that $F(\alpha) = 0$ and $\alpha \equiv \gamma \pmod{p}$.

Here is an example.

```
In[32]:= PadicNewton[x^2 - 1, x, 2, 0.0001, 3]
```

```
Out[32]= <3> 2 . 2 2 2 2 2 2 2
```

■ References

- [1] W. Schikhof, *Ultrametric Calculus: An Introduction to p-adic Analysis*, New York: Cambridge University Press, 1984.
- [2] N. Koblitz, *p-adic Numbers, p-adic Analysis, and Zeta-Functions*, London: Springer-Verlag, 1977.
- [3] S. Wolfram, *The Mathematica Book*, 5th ed., Champaign: Wolfram Media, Inc., 2003.

■ Additional Material

PadicArithmetic.m

Available at www.mathematica-journal.com.

Stany De Smedt
KBC Bank & Verzekeringen
Havenlaan 12, 1080 Brussels, Belgium

Updating a Geographic Database

A Rubber-Sheet Adjustment Algorithm

Leendert van Gastel
Harry Uitermark

A rubber-sheet algorithm for warping is used to update a geographic database without altering points that are already known in the database. *Mathematica* allows a quick experimental validation of the theory and of existing FORTRAN legacy code.

■ Introduction

The number of geographic databases is increasing rapidly in many domains, varying from municipal management to tourism and archeology: anyone managing a territory needs their own geographic database [1].

Geographic databases form the data part of geographic information systems (GIS). At the Dutch Cadastre (Land Registration) we have built our own GIS, which is primarily a geographic database with houses and ownership boundaries.

In the beginning, paper maps were digitized as the data source for this geographic database. Now that all existing paper maps are in the database, it is important to keep the database up to date. For example, when a new house is built, the database will be updated (Figure 1). Central to this updating is that we do not want to change the coordinates of the information that is already in the geographic database.

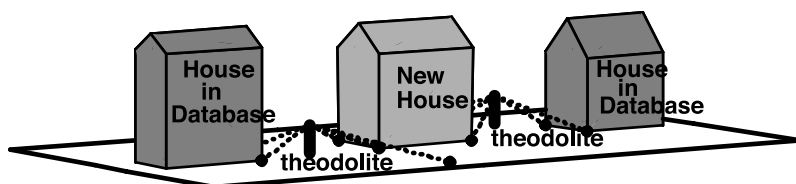


Figure 1. Measuring a new house (the green one) using a theodolite from two different positions.

However, we sometimes did not trust the results of existing update programs, which were coded in FORTRAN. We wondered if our formulae were correctly implemented, or were wrong from the beginning. So we checked our formulae by implementing them in *Mathematica* before checking the FORTRAN code. In the end the formulae was proven right and we found a bug in the FORTRAN code.

In this article we use this implementation to demonstrate how new information is added to the geographic database.

■ Warping

When collecting new information—most of the time using a theodolite, which is a surveying instrument that measures bearings and distances—all observations are done in a redundant way. This redundancy makes it possible to check the observations for errors. But at the same time, due to the stochastic nature of measurements, small discrepancies are introduced. So before adding new information to the database, we check our measurements for errors and filter out the small discrepancies by least squares methods. A statistical test is performed to ensure the quality of the data: the covariance of the measurements should be within standardized bounds.

A second stage, called *warping*, fits this internal local computation into the external global world of the database. There are several methods in use for warping. We mention:

- A similarity transformation, where the coefficients are determined by interpolation
- A quadratic bilinear extension
- An elastic rubber-sheet transformation [2]

At the Dutch Cadastre, we chose the last method because it deals best with the information already in the database. We will illustrate this later with an example.

■ Rubber-Sheet Adjustment

When adding a new measurement to the database, we discriminate between control points and free points.

Control points are points that already exist in the database; free points are new points to be added to the database. As we see in Figure 2, the points pt3, pt4, pt30, pt31, and pt504 are already in the database and therefore they are control points. The points pt20, pt21, pt25, and pt32 are from the new house and therefore free. Points pt1 and pt2 are the theodolite station points. These points are used in the computation but not stored in the database.

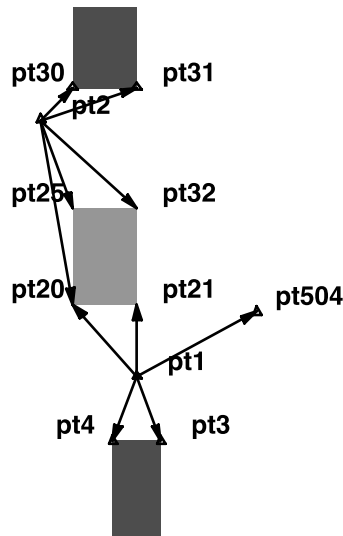


Figure 2. The control points (pt3, pt4, pt30, and pt504) and free points (pt20, pt21, pt25, and pt32) of the database.

The rubber-sheet transformation maps the measured coordinates of the control points exactly onto the coordinates of the control points in the database (this is the prime reason for choosing the rubber-sheet transformation).

There are two main steps in the operation.

- The first step is to link the coordinate system of the measurement with the database. This is done by a similarity transformation. This transformation can only map the measured coordinates of two control points exactly onto the coordinates in the database. So, there are differences for the other control points.
- The second step is to eliminate these differences.

The following formula is central to the second step:

$$e_f = H_{fc} \cdot H_{cc}^{-1} \cdot e_c \quad (1)$$

with

- e_f being the corrections to the coordinates of a free point
- H_{fc} representing the covariance matrix of a free point and the control points
- H_{cc} representing the covariance matrix of the control points
- e_c being the vector of differences between the coordinates of the control points in the measurement (after the similarity transformation) and the database.

This formula expresses the correction for any point as a weighted linear combination of the corrections of the control points. The weights depend on the distance

between the point and the control points. We explain below how the matrices H_{fc} and H_{cc} are constructed.

The most important property of this method is that it does not matter which basis you choose for the similarity transformation: after applying the above formula, you get the same result regardless of the basis. This is a form of invariance.

■ The Similarity Transformation

First, two control points r and s are chosen. These two points are mapped precisely onto the database by the similarity transformation S . A similarity transformation is by definition composed of rotation, scaling, and translation, so the transformation S is given by

$$x^{(2)} = Sx^{(1)} = \begin{pmatrix} p & -q \\ q & p \end{pmatrix} (x^{(1)} - r^{(1)}) + r^{(2)},$$

where p and q are found by solving $s^{(2)} = Ss^{(1)}$. Here $x^{(2)}$ denotes the coordinates of the point in the database and $x^{(1)}$ denotes the coordinates of the point in the measurement.

The *Mathematica* implementation is straightforward.

```
In[1]:= A = {{p, -q}, {q, p}};
r1 = {r1x, r1y};
r2 = {r2x, r2y};
s1 = {s1x, s1y};
s2 = {s2x, s2y};
S[{x_, y_}] := A . ({x, y} - r1) + r2;
```

The numbers p and q are found by the following.

```
In[7]:= sols = Evaluate[Solve[S[s1] == s2, {p, q}]]][[1]]
```

```
Out[7]= {p -> -(-r1x r2x - r1y r2y + r2x s1x + r2y s1y + r1x s2x - s1x s2x + r1y s2y -
s1y s2y) / (r1x^2 + r1y^2 - 2 r1x s1x + s1x^2 - 2 r1y s1y + s1y^2),
q -> -(-r1y + s1y) (r2x - s2x) + (-r1x + s1x) (r2y - s2y) /
(r1x + s1x)^2 - (r1y - s1y) (-r1y + s1y)}
```

■ The Covariance Matrix

For two sets of points $\{u_1, \dots, u_m\}$ and $\{v_1, \dots, v_n\}$, the covariance matrix is made from 2×2 blocks:

$$H_{\{u_i\}\{v_j\}}^{r,s} = \begin{pmatrix} b_{xx}(u_1, v_1) & b_{xy}(u_1, v_1) & \dots & b_{xx}(u_1, v_n) & b_{xy}(u_1, v_n) \\ -b_{xy}(u_1, v_1) & b_{xx}(u_1, v_1) & \dots & -b_{xy}(u_1, v_n) & b_{xx}(u_1, v_n) \\ \vdots & & & & \\ b_{xx}(u_m, v_1) & b_{xy}(u_m, v_1) & \dots & b_{xx}(u_m, v_n) & b_{xy}(u_m, v_n) \\ -b_{xy}(u_m, v_1) & b_{xx}(u_m, v_1) & \dots & -b_{xy}(u_m, v_n) & b_{xx}(u_m, v_n) \end{pmatrix}$$

where r and s denote a basis and

$$b_{xx}(u, v) = -|u - v| + (1 - a(v))|u - r| + a(v)|u - s| + (1 - a(u))|u - r| \\ + a(u)|v - s| + (-a(u) - a(v) + 2a(u)a(v) + 2b(u)b(v))|s - r|$$

and

$$b_{xy}(u, v) = b(v)|u - r| - b(v)|u - s| - b(u)|v - r| + b(v)|v - r| \\ + b(u)|v - s| - (b(u) - b(v) - 2a(v)b(u) + 2a(u)b(v))|s - r|$$

with

$$a(u) = (u - r) \cdot (s - r) / (s - r) \cdot (s - r) \\ b(u) = (u - r)^\top \cdot (s - r) / (s - r) \cdot (s - r).$$

In equation (1), the term H_{f_c} is the matrix H for a single free point f and the set of control points c . The term H_{c_c} is the matrix H for both sets of control points c .

■ Mathematical Background

For the covariance matrix, we do not take the actual covariance of this particular measurement, but start from a standardized covariance matrix whose coefficients depend only on the distances between points. This has the advantage that the warping does not depend on the particular measurement. This choice is allowed because we know that the measurements are sufficiently reliable since they have passed a standardized test. In fact, the specifications on the covariance in this test are reflected in the new matrix.

The matrix is constructed in several steps. Let us say that the vicinity of two coordinate values x and x' is measured by

$$d(x, x') := D - (x - x')^2$$

for some fixed positive number D . If $d(x, x')$ would be negative, we set it equal to 0. The standardized covariance matrix for two points (x, y) and (x', y') has the form

$$\begin{pmatrix} d(x, x') & 0 \\ 0 & d(y, y') \end{pmatrix}.$$

This tells us that if two points are close, then the x -coordinates and the y -coordinates are close, but there is *a priori* no cross relation between the x -coordinate and the y -coordinate. For two sets of points $\{u_1, \dots, u_m\}$ and $\{v_1, \dots, v_n\}$ the matrix is just made from 2×2 blocks by taking as i, j -th block the covariance matrix of u_i and v_j :

$$\begin{pmatrix} d(u_{1,x}, v_{1,x}) & 0 & \dots & d(u_{1,x}, v_{n,x}) & 0 \\ 0 & d(u_{1,y}, v_{1,y}) & \dots & 0 & d(u_{1,y}, v_{n,y}) \\ \vdots & & & & \\ d(u_{m,x}, v_{1,x}) & 0 & \dots & d(u_{m,x}, v_{n,x}) & 0 \\ 0 & d(u_{m,y}, v_{1,y}) & \dots & 0 & d(u_{m,y}, v_{n,y}) \end{pmatrix}.$$

The above is the standard covariance matrix in the local coordinates of the measurement. During the warping we use the coordinate system of the database, so the matrix is affected by the similarity transformation and the result is $H_{\{u_i\}\{v_j\}}^{r,s}$. This is worked out in [2], where you may find more details.

■ Implementation

Here is the implementation of the rubber-sheet transformation.

```

In[8]:= a[x_, r_, s_] := (r - x).(r - s)/(r - s).(r - s);
b[x_, r_, s_] := T[r - x).(r - s)/(r - s).(r - s);
T[{a_, b_}] := {-b, a};
d2[x_List, y_List] := N[Sqrt[(x - y).(x - y)]];

hxx[i_List, j_List, r_, s_] :=
  -d2[i, j] - (-1 + a[j, r, s])*d2[i, r] + a[j, r, s]*d2[i, s] -
  (-1 + a[i, r, s])*d2[j, r] + a[i, r, s]*d2[j, s] +
  d2[r, s]*(-a[i, r, s] - a[j, r, s] + 2*a[i, r, s]*a[j, r, s] +
  2*b[i, r, s]*b[j, r, s]);

hxy[i_List, j_List, r_, s_] :=
  b[j, r, s]*d2[i, r] - b[j, r, s]*d2[i, s] - b[i, r, s]*d2[j, r] +
  b[i, r, s]*d2[j, s] - (b[i, r, s] - 2*a[j, r, s]*b[i, r, s] -
  b[j, r, s] + 2*a[i, r, s]*b[j, r, s])*d2[r, s];

h[p_, q_, r_, s_] := {{hxx[p, q, r, s], hxy[p, q, r, s]},
  {-hxy[p, q, r, s], hxx[p, q, r, s]}};

covarianceBlock[h_, u_, {v__}, r_, s_] := Module[{x}, Flatten/@
  Thread[Apply[h[#1, #2, r, s] &, Thread[{x, {v}}], {1}]] /. x -> u];

covarianceMatrix[h_, u_List, {v__}, r_, s_] :=
  Flatten[covarianceBlock[h, #, {v}, r, s] & /@ u, 1];

covarianceMatrix[h_, u_, {v__}, r_, s_] :=
  covarianceMatrix[h, {u}, {v}, r, s];

```

The definition of `covarianceBlock`, which makes a row of 2×2 blocks as in H_{fc} , is probably the only one that may need some explanation. For example, let us make the matrix for `u` and `{v1, v2}` with the help of `Thread`.

```

In[18]:= Thread[{u, {v1, v2}}]

```

```

Out[18]= {{u, v1}, {u, v2}}

```

Next we apply `h` on the two blocks at the right level.

```

In[19]:= Apply[h[#1, #2, r, s] &, %, {1}]

```

```

Out[19]= {{hxx[u, v1, r, s], hxy[u, v1, r, s]}, {-hxy[u, v1, r, s], hxx[u, v1, r, s]}},
  {{hxx[u, v2, r, s], hxy[u, v2, r, s]}, {-hxy[u, v2, r, s], hxx[u, v2, r, s]}}

```

Finally we flatten to get a matrix.

In[20]:= **Flatten** /@ %

```
Out[20]= {{hxx[u, v1, r, s], hxy[u, v1, r, s],
          -hxy[u, v1, r, s], hxx[u, v1, r, s]}, {hxx[u, v2, r, s],
          hxy[u, v2, r, s], -hxy[u, v2, r, s], hxx[u, v2, r, s]}}
```

To make this code work for a concrete vector (u_x, u_y) , as well as for a symbol u , we first introduce a free variable x instead of u and then use the substitution $x \rightarrow u$. This variable has been made local with the use of `Module`. All together we obtain `covarianceBlock`.

■ Example

Here is the example of Figure 2. Table 1 gives the measured coordinates and the coordinates of the database. First we choose pt3 and pt504 as a basis for the similarity transformation. Then we show that another basis, for example pt30 and pt31, will give the same result for the final coordinates of a measurement.

	Measured coordinates		Database coordinates	
	x	y	x	y
pt20	1992.000	2009.000	to	to
pt21	2000.000	2009.000	be	be
pt25	1991.997	2021.000	com-	com-
pt32	2000.001	2021.001	puted	puted
pt3	2003.000	1992.000	155003.032	462992.079
pt4	1997.007	1991.998	154997.052	462992.039
pt30	1991.994	2036.000	154992.035	463036.095
pt31	1999.994	2036.002	155000.085	463036.052
pt504	2015.000	2008.000	155015.041	463008.000

Table 1. Measured coordinates and database coordinates.

We choose $r = \text{pt3}$ and $s = \text{pt504}$ as a basis. The other control points are the points pt4, pt30, and pt31. We transform the local coordinates of the control points and the free points (pt20, pt21, pt25, and pt32) with respect to the chosen basis. Here we only work out pt20.

```
In[21]:= r1x = 2003; r1y = 1992; r2x = 155003.032; r2y = 462992.079;
s1x = 2015; s1y = 2008; s2x = 155015.041; s2y = 463008.000;
pt20 = {1992, 2009};
pt21 = {2000, 2009};
pt25 = {1991.997, 2021.000};
pt32 = {2000.001, 2021.001};
pt20t = S[pt20] /. sols;
```

```

pt21t = S[pt21] /. sols;
pt25t = S[pt25] /. sols;
pt32t = S[pt32] /. sols;
pt4 = {1997.007, 1991.998};
pt30 = {1991.994, 2036};
pt31 = {1999.994, 2036.002};
pt4t = S[pt4] /. sols;
pt30t = S[pt30] /. sols;
pt31t = S[pt31] /. sols;
pt3db = {r2x, r2y};
pt4db = {154997.052, 462992.039};
pt30db = {154992.035, 463036.095};
pt31db = {155000.085, 463036.052};
pt504db = {s2x, s2y};

```

We transform the local coordinates of the control points and the free points (pt20, pt21, pt25, and pt32) with respect to the chosen basis. Here we only work out pt20.

```
In[42]:= pt20t = S[pt20] /. sols // N[#, 10] &
```

```
Out[42]= {154992., 463009.}
```

Next we compute the database coordinates of the new house with the following formula.

```

In[43]:= hfc = covarianceMatrix[h,
    {pt20t, pt21t, pt25t, pt32t},
    {pt4t, pt30t, pt31t}, pt3db, pt504db];
hcc = covarianceMatrix[h,
    {pt4t, pt30t, pt31t},
    {pt4t, pt30t, pt31t}, pt3db, pt504db];
ec = Flatten[{pt4db, pt30db, pt31db}] -
    Flatten[{pt4t, pt30t, pt31t}];
ef = hfc.Inverse[hcc].ec

```

```
Out[46]= {-0.0686549, 0.000513474, -0.0423258, 0.00651125,
    -0.0968291, 0.044021, -0.0634426, 0.0439568}
```

Here are the final coordinates of pt20, pt21, pt25, and pt32.

```
In[47]:= def3504 = Flatten[{pt20t, pt21t, pt25t, pt32t}] + ef // N[#, 10] &
```

```
Out[47]= {154992., 463009., 155000.,
    463009., 154992., 463021., 155000., 463021.}
```

Table 2 summarizes the result.

	Transformed		Database		Difference	
	<i>x</i>	<i>y</i>	<i>x</i>	<i>y</i>	<i>x</i>	<i>y</i>
pt20	154992.110	463009.060	154992.042	463009.060	-0.069	0.001
pt21	155000.087	463009.038	155000.045	463009.045	-0.042	0.007
pt25	154992.140	463021.025	154992.043	463021.069	-0.097	0.044
pt32	155000.121	463021.004	155000.057	463021.048	-0.063	0.044
pt3	155003.032	462992.079	155003.032	462992.079	0.000	0.000
pt4	154997.056	462992.093	154997.052	462992.039	-0.004	-0.054
pt30	154992.178	463035.982	154992.035	463036.095	-0.143	0.113
pt31	155000.155	463035.962	155000.085	463036.052	-0.070	0.090
pt504	155015.041	463008.000	155015.041	463008.000	0.000	0.000

Table 2. The transformed coordinates with respect to pt3 and pt504.

■ **Another Basis**

Choosing another basis for the similarity transformation, for example $r = \text{pt30}$ and $s = \text{pt31}$, leads to an entirely different computation, but the final results will be the same.

Again Table 1 is our starting point. Now the other control points are the points pt3, pt4, and pt504. We transform the local coordinates of these control points and the free points (pt20, pt21, pt25, and pt32) with respect to this basis. The results are in a vector `def3031`. Table 3 shows the results.

	Transformed		Database		Difference	
	<i>x</i>	<i>y</i>	<i>x</i>	<i>y</i>	<i>x</i>	<i>y</i>
pt20	154991.889	463008.926	154992.042	463009.060	0.152	0.134
pt21	154999.939	463008.881	155000.045	463009.045	0.106	0.163
pt25	154991.954	463021.001	154992.043	463021.069	0.090	0.068
pt32	155000.008	463020.957	155000.057	463021.048	0.050	0.091
pt3	155002.862	462991.758	155003.032	462992.079	0.170	0.321
pt4	155002.862	462991.758	154997.052	462992.039	0.220	0.249
pt30	154992.035	463036.095	154992.035	463036.095	0.000	0.000
pt31	155000.085	463036.052	155000.085	463036.052	0.000	0.000
pt504	155015.027	463007.791	155015.041	463008.000	0.014	0.209

Table 3. The transformed coordinates with respect to pt30 and pt31.

We redo the computation for this case.

```

In[48]:= r1x=1991.994; r1y=2036.000; r2x=154992.035; r2y=463036.095;
s1x=1999.994; s1y=2036.002; s2x=155000.085; s2y=463036.052;
pt20t=S[pt20] /. sols;
pt21t=S[pt21] /. sols;
pt25t=S[pt25] /. sols;
pt32t=S[pt32] /. sols;
pt4={1997.007, 1991.998};
pt3 = {2003, 1992};
pt504 = {2015, 2008};
pt4t= S[pt4] /. sols;
pt3t= S[pt3] /. sols;
pt504t= S[pt504] /. sols;
pt3db={155003.032, 462992.079};
pt4db = {154997.052, 462992.039};
pt30db={r2x, r2y};
pt31db={s2x, s2y};
pt504db={155015.041, 463008.000};

hfc = covarianceMatrix[h,
{pt20t, pt21t, pt25t, pt32t}, {pt3t, pt4t, pt504t}, pt30db, pt31db];

hcc = covarianceMatrix[h,
{pt3t, pt4t, pt504t}, {pt3t, pt4t, pt504t}, pt30db, pt31db];

ec = Flatten[{pt3db, pt4db, pt504db}] - Flatten[{pt3t, pt4t, pt504t}];

In[68]:= ef = hfc.Inverse[hcc].ec

Out[68]= {0.152425, 0.134159, 0.105645, 0.163329,
0.0895193, 0.067995, 0.0497575, 0.0911056}

Here are the final coordinates of pt20, pt21, pt25, and pt32 with respect to the
basis pt30 and pt31.

In[69]:= def3031 =
N[Flatten[{pt20t, pt21t, pt25t, pt32t}]+ ef, 10]

Out[69]= {154992., 463009., 155000.,
463009., 154992., 463021., 155000., 463021.}

```

■ Visualizing Results

To visualize and compare the results of the different bases, we take a corner of the house and show the images of the corner under the two similarity transformations together with the final result of both computations (Figure 3).

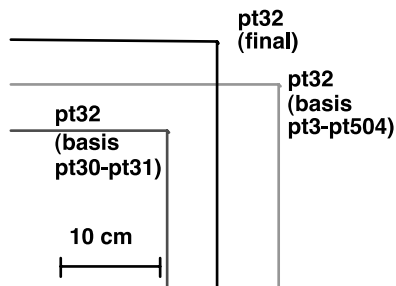


Figure 3. A zoomed-in picture of a corner of the new house, where you can see the effect of the different choices.

In Figure 3 the green corner is the image under the first transformation, the red corner is the image under the second transformation, and the black corner is the final result of both computations.

We can also perform a numerical check: the final results of the second computation should be numerically equal to those of the first computation.

```
In[70]:= def3504 - def3031
```

```
Out[70]:= {-3.7835 × 10-10, 2.32831 × 10-10, -4.65661 × 10-10, -1.33878 × 10-9,  
          9.31323 × 10-10, 1.16415 × 10-10, 2.44472 × 10-9, -1.28057 × 10-9}
```

We can make a picture of the warping by applying `PlotVectorField` from the package `Graphics`PlotField``. The vector field has been combined with the map using `Show`. Figure 4 shows the result: the “vector field” is like a vortex with two “sinks” at the basis points `pt30` and `pt31` where the corrections should be zero.

```
In[71]:= << Graphics`PlotField`
```

```
In[72]:= f = covarianceMatrix[h, {{x, y}}, {pt3t, pt4t, pt504t}, pt30db, pt31db].  
          Inverse[hcc].ec;
```

```
In[73]:= PlotVectorField[f, {x, pt4t[[1]] - 20, pt504t[[1]] + 5, 10},
  {y, pt3t[[2]] - 15, pt31db[[2]] + 15, 10}]
```

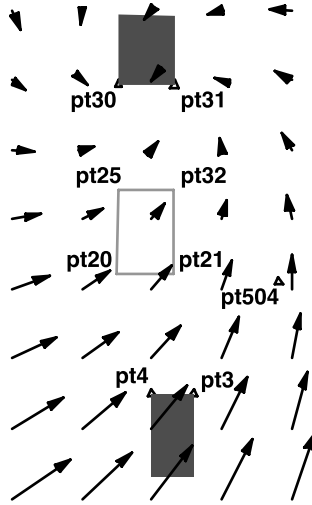


Figure 4. A “vector field” showing the relative size and direction of the corrections to the transformed coordinates.

■ Conclusion

The rubber-sheet algorithm for warping allows us to update a geographic database without altering points that are already known in the database. *Mathematica* offers a quick experimental validation of the theory and of the existing FORTRAN legacy code.

■ References

- [1] R. Laurini and D. Thompson, *Fundamentals of Spatial Information Systems*, London: Academic Press, 1992.
- [2] H. J. Buiten and P. Richardus, *Junction of Control Surveys by Adjustment Compared with Coordinate Transformation*, München: Hochschule der Bundeswehr, Schriftenreihe Wissenschaftlicher Studiengang Vermessungswesen, Heft 7, 1982, pp. 115–141.

About the Authors

Leendert van Gastel is a consultant specializing in the use of ICT in science and mathematics. He has been working with *Mathematica* since 1992, both in applied research and training.

Harry Uitermark is a GIS consultant. Currently his research interests include geographic information, ontologies, and XML Schema, a web standard for describing the structure and semantics of XML documents.

Leendert van Gastel

Amstel Instituut

University of Amsterdam

gastel@science.uva.nl

Harry Uitermark

Kadaster, Box 9046

Apeldoorn 7300 GH, The Netherlands

harry.uitermark@kadaster.nl

A Generator of Rook Polynomials

Daniel C. Fielder

A list adaptation of an inclusion-exclusion method for calculating the rook polynomials of arbitrary finite chessboards is discussed and presented.

■ General

On a chessboard of any configuration, a rook is *nontaking* if the cell it occupies is not in the same row or column as the cell of any other rook. A *rook polynomial* is an ordinary generating function in which the coefficient of x^k ($k = 0, 1, 2, \dots$) is the number of ways k nontaking rooks can be distributed on a finite board. Rook polynomials have many applications in combinatorics, especially in counting restricted permutations. A chessboard can be described mathematically as an ordered set of its cells' unique (row, column) pairs. A chessboard need not be rectangular. The algorithm for finding rook polynomials discussed in this article is an adaptation of a well-known inclusion-exclusion method, typically described by Riordan [1, 168–170] and Liu [2, 111–118].

This method selects any cell of the board as a *special cell*. As Liu observes, the number of ways k nontaking rooks on the original chessboard C is the number of ways with a rook always *included* in the special cell plus the number of ways with all rooks *excluded* from the special cell. The board C_i is found by crossing out the row *and* column of the special cell. The contribution of the rook polynomial from C_i must be multiplied by x to account for the loss of the cell along with its row and column. The board C_e is found by eliminating the special cell completely. Hence, $C = x C_i + C_e$ forms an interesting “equation” in x , with chessboards as coefficients. Continued reductions of C_i and C_e and their successor boards yield nested sets of x 's from which the rook polynomial of board C emerges. When working “by hand,” recognition of the rook polynomials of intermediate boards often shortens this search—a luxury usually not afforded in a computer reduction.

■ A “By Hand” Example

This small example provides a convenient comparison to the computer algorithm adaptation later. Consider the following chessboard of four cells and (arbitrarily)

make $\begin{bmatrix} 1, 1 \end{bmatrix}$ the special cell at the start. The choice of intermediate special cells is obvious. The following reduction leads to the unique rook polynomial, $1 + 4x + 3x^2$.

$$\begin{array}{|c|c|c|} \hline 1, 1 & & \\ \hline 2, 1 & 2, 2 & \\ \hline & 3, 2 & \\ \hline \end{array} = x \begin{array}{|c|c|} \hline 2, 2 & \\ \hline 3, 2 & \\ \hline \end{array} + \begin{array}{|c|c|c|} \hline 2, 1 & 2, 2 & \\ \hline & 3, 2 & \\ \hline \end{array} = x(x \begin{array}{|c|} \hline \\ \hline \end{array} + \begin{array}{|c|} \hline 3, 2 \\ \hline \end{array}) + \left(x \begin{array}{|c|} \hline 3, 2 \\ \hline \end{array} + \begin{array}{|c|c|} \hline 2, 2 & \\ \hline 3, 2 & \\ \hline \end{array} \right) =$$

$$x^2 + 2x \begin{array}{|c|} \hline 3, 2 \\ \hline \end{array} + \left(x \begin{array}{|c|} \hline \\ \hline \end{array} + \begin{array}{|c|} \hline 3, 2 \\ \hline \end{array} \right) = x^2 + x + (2x + 1) \begin{array}{|c|} \hline 3, 2 \\ \hline \end{array} =$$

$$x^2 + x + (2x + 1)(x + 1) = x^2 + x + 2x + 2x^2 + 1 + x = 1 + 4x + 3x^2$$

The steps taken use algebraic combination and the knowledge that the rook polynomials of the empty chessboard $\begin{array}{|c|} \hline \\ \hline \end{array}$ and a single cell (in this case $\begin{array}{|c|} \hline 3, 2 \\ \hline \end{array}$) are 1 and $1 + x$.

■ A List Example

In this example, the computer algorithm for finding rook polynomials is an orderly list adaptation of the “by hand” example. First, initiate a horizontal “stack” with the list of the chessboard’s (row, column) cells on the initial top of the stack. (In the implementation the top will be the first element of the list `stacklist`.) From data in the top of the stack, generate a list of cells of the first *exclusive* type board and append it to the bottom of the stack. (In the implementation the bottom will be put farthest to the right). Next, compare the (row, column) pattern information of the first cell of the top with the patterns of the remaining top cells to obtain the cell information for the first *inclusive* type list. The x multiplier has the same functional weight as a (row, column) cell in the inclusive list and is added on the right to complete the inclusive list. The inclusive list is appended to the right of the exclusive list and is now at the bottom of the stack.

Here is the result of applying the process to the first top, which is the first element in this list.

$$\{\{1, 1\}, \{2, 1\}, \{2, 2\}, \{3, 2\}\}, \{\{2, 1\}, \{2, 2\}, \{3, 2\}\}, \{\{2, 2\}, \{3, 2\}, \{x\}\}$$

In the following table this will look like a stack.

$$\begin{array}{c} \{\{1, 1\}, \{2, 1\}, \{2, 2\}, \{3, 2\}\} \\ \{\{2, 1\}, \{2, 2\}, \{3, 2\}\} \\ \{\{2, 2\}, \{3, 2\}, \{x\}\} \end{array}$$

The top of the stack has served its purpose and is discarded. The lists can be envisioned as propagating up with each removal of a used top (or to the left in the list implementation). The general philosophy is to repeat the generation and removal of new tops until the stack consists only of x entries and a single 1. The single 1 is derived from the *exclusive* reduction of a single cell (see [2, 113] and the earlier use of $\begin{array}{|c|} \hline 3, 2 \\ \hline \end{array} = 1 + x$).

The following tabulation shows the progress of the algorithm through the 17 successive stack values.

1	$\{\{1, 1\}, \{2, 1\}, \{2, 2\}, \{3, 2\}\}$	2	$\{\{1, 1\}, \{2, 1\}, \{2, 2\}, \{3, 2\}\}$ $\{\{2, 1\}, \{2, 2\}, \{3, 2\}\}$ $\{\{2, 2\}, \{3, 2\}, \{x\}\}$	3	$\{\{2, 1\}, \{2, 2\}, \{3, 2\}\}$ $\{\{2, 2\}, \{3, 2\}, \{x\}\}$
4	$\{\{2, 1\}, \{2, 2\}, \{3, 2\}\}$ $\{\{2, 2\}, \{3, 2\}, \{x\}\}$ $\{\{2, 2\}, \{3, 2\}\}$ $\{\{3, 2\}, \{x\}\}$	5	$\{\{2, 2\}, \{3, 2\}, \{x\}\}$ $\{\{2, 2\}, \{3, 2\}\}$ $\{\{3, 2\}, \{x\}\}$	6	$\{\{2, 2\}, \{3, 2\}, \{x\}\}$ $\{\{2, 2\}, \{3, 2\}\}$ $\{\{3, 2\}, \{x\}\}$ $\{\{3, 2\}, \{x\}\}$ $\{\{x\}, \{x\}\}$
7	$\{\{2, 2\}, \{3, 2\}\}$ $\{\{3, 2\}, \{x\}\}$ $\{\{3, 2\}, \{x\}\}$ $\{\{x\}, \{x\}\}$	8	$\{\{2, 2\}, \{3, 2\}\}$ $\{\{3, 2\}, \{x\}\}$ $\{\{3, 2\}, \{x\}\}$ $\{\{x\}, \{x\}\}$ $\{\{3, 2\}\}$ $\{\{x\}\}$	9	$\{\{3, 2\}, \{x\}\}$ $\{\{3, 2\}, \{x\}\}$ $\{\{x\}, \{x\}\}$ $\{\{3, 2\}\}$ $\{\{x\}\}$
10	$\{\{3, 2\}, \{x\}\}$ $\{\{3, 2\}, \{x\}\}$ $\{\{x\}, \{x\}\}$ $\{\{3, 2\}\}$ $\{\{x\}\}$ $\{\{x\}\}$ $\{\{x\}, \{x\}\}$	11	$\{\{3, 2\}, \{x\}\}$ $\{\{x\}, \{x\}\}$ $\{\{3, 2\}\}$ $\{\{x\}\}$ $\{\{x\}\}$ $\{\{x\}, \{x\}\}$	12	$\{\{3, 2\}, \{x\}\}$ $\{\{x\}, \{x\}\}$ $\{\{3, 2\}\}$ $\{\{x\}\}$ $\{\{x\}\}$ $\{\{x\}, \{x\}\}$ $\{\{x\}\}$ $\{\{x\}, \{x\}\}$
13	$\{\{x\}, \{x\}\}$ $\{\{3, 2\}\}$ $\{\{x\}\}$ $\{\{x\}\}$ $\{\{x\}, \{x\}\}$ $\{\{x\}\}$ $\{\{x\}, \{x\}\}$	14	$\{\{3, 2\}\}$ $\{\{x\}\}$ $\{\{x\}\}$ $\{\{x\}, \{x\}\}$ $\{\{x\}\}$ $\{\{x\}, \{x\}\}$ $\{\{x\}, \{x\}\}$	15	$\{\{3, 2\}\}$ $\{\{x\}\}$ $\{\{x\}\}$ $\{\{x\}, \{x\}\}$ $\{\{x\}\}$ $\{\{x\}, \{x\}\}$ $\{\{x\}, \{x\}\}$ $\{\{1\}\}$ $\{\{x\}\}$
16	$\{\{x\}\}$ $\{\{x\}\}$ $\{\{x\}, \{x\}\}$ $\{\{x\}\}$ $\{\{x\}, \{x\}\}$ $\{\{x\}, \{x\}\}$ $\{\{1\}\}$ $\{\{x\}\}$	17	$\{\{1\}\}$ $\{\{x\}\}$ $\{\{x\}\}$ $\{\{x\}\}$ $\{\{x\}, \{x\}\}$ $\{\{x\}\}$ $\{\{x\}, \{x\}\}$ $\{\{x\}, \{x\}\}$		

The stacks 1, 3, 5, 7, 9, 11, 14, and 17 are stable stack configurations, while the others are stacks in transition. In particular, when top lists consist solely of x 's (as in stacks 13 and 16), those lists are transferred to the bottom of the stack before normal stack operation resumes. The algebraic interpretation of the final stack is to multiply along the rows and add up the column, ignoring the list structure. $\{\{1\}\}$ is interpreted as 1. Under these conditions, the program will convert stack 17 to the rook polynomial, $1 + 4x + 3x^2$.

■ Code and Program Details

Here is the program for converting a list of nonrepeating (row, column) chessboard cells to a rook polynomial and its application to our small example.

```
In[2]:= RookPolynomial[n_List] :=
Module[{stacklist, top, SameColumnValues, SameRowValues,
  EntriesToEliminate, InclusionList, ExclusionList},
  stacklist = {Union[n]};
  While[Union[Flatten[stacklist]] != {1, x}, top = First[stacklist];
    SameColumnValues = Position[top, {_, top[[1, 2]]}];
    SameRowValues = Position[top, {top[[1, 1]], _}];
    EntriesToEliminate = Union[SameColumnValues, SameRowValues];
    InclusionList = Append[Delete[top, EntriesToEliminate], {x}];
    ExclusionList = Delete[top, 1];
    If[Length[ExclusionList] == 0, ExclusionList = {{1}}];
    stacklist = Delete[
      Append[Append[stacklist, ExclusionList], InclusionList], 1];
    While[Union[Flatten[First[stacklist]]] == {x},
      stacklist = RotateLeft[stacklist]];
    Total[Apply[Times, stacklist, 2]]]
```

```
In[3]:= RookPolynomial[{{2, 2}, {3, 2}, {3, 3}, {4, 3}}]
```

```
Out[3]= 1 + 4 x + 3 x^2
```

When `stacklist` reaches $\{\{1\}\}, \{x\}, \dots, \{x\}$ under `While` loop control, `stacklist` has reached its correct, final value. This is the only value that yields the rook polynomial directly. Also, `Union[Flatten[stacklist]]` equals $\{1, x\}$ only for the final `stacklist` value.

`SameColumnValues` uses the pattern properties of the `Position` command to obtain a list of positions from the top of the stack that have the same *column* values as the leading (row, column) entry at the top of the stack list. `SameRowValues` does the same for the *row* values. Their `Union` gives the list of (row, column) `EntriesToEliminate` in forming the next *inclusion* chessboard, which is completed by appending x . The *exclusion* chessboard list is completed by deleting the first term from the top of the stack. After that, the inclusion and exclusion chessboard strings are appended to the bottom of the stack before the old top of the stack is discarded.

The lists move up (actually over to the left) and the next list becomes the new top of the stack. If not prevented, the program will try to process any stack tops that have leading x 's. Accordingly, the top of a stack list consisting entirely of x 's is moved to the bottom of the stack in the last `While` loop. When `stacklist` eventually meets the condition for shutting down the main `While` loop, the `Apply` line converts `stacklist` to the rook polynomial.

In this manner, `RookPolynomial[{{row,column},...,{row,column}}]` can be determined for any connected chessboard.

■ References

- [1] J. Riordan, *An Introduction to Combinatorial Analysis*, New York: Wiley, 1958.
- [2] C. Liu, *Introduction to Combinatorial Mathematics*, New York: McGraw-Hill, 1968.

About the Author

Daniel C. Fielder received a Ph.D. in electrical engineering from Georgia Institute of Technology in 1957. Except for a year of teaching at Syracuse University and a summer visiting professorship at Westinghouse Research Labs, he taught continuously and performed research on discrete math applications at Georgia Tech since 1948. He was a Life Fellow of IEEE. Dr. Fielder retired in 1988 as professor emeritus, but continued to teach until the time of his death in October 2003. During WWII, he designed ship degaussing systems at the Bureau of Ships under (then) Lt. Cmdr. Hyman G. Rickover, who later founded the U.S. nuclear navy.

Daniel C. Fielder

*School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332-0250*

Customizing the Help Browser

Pavi Sandhu

■ Introduction

The Help Browser provides convenient access to all of *Mathematica*'s online documentation. You can browse through a hierarchy of topics or search for information on a specific keyword. You can also edit and evaluate the examples or print any topic displayed in the Help Browser, just like in a notebook.

A little-known feature of the Help Browser is that you can customize it. You can add your own reference material to the existing help files, add your own keywords to the Master Index, or set up hyperlinks between different categories of the Help Browser. You can even change the names and number of the buttons at the top of the Help Browser

These types of changes are useful if you are creating an add-on package or a custom application. You can then integrate your documentation into the Help Browser. This article provides detailed information on the structure of the Help Browser's files and the steps needed to modify them.

All information to be displayed in the Help Browser must be stored in the form of notebooks inside the Documentation directory in the *Mathematica* layout. The Documentation directory can contain several subdirectories, each with the name of a specific language. For example, all documentation in English should be placed in the Documentation/English directory.

The Documentation/English directory in turn contains five subdirectories, each corresponding to a specific button in the Help Browser:

- AddOns
- Demos
- GettingStarted
- MainBook
- OtherInformation
- RefGuide
- Tour

In addition to the notebooks containing the help material, each of these directories must contain its own copy of the following two files:

- `BrowserCategories.m`—This is a plain text file that specifies the categories to be displayed in the Browser window.
- `BrowserIndex.nb`—This is a notebook that specifies the entries that should appear in the Master Index.

To add new reference material to the Help Browser:

1. Place the notebooks containing the new material inside the `AddOns` subdirectory of the `Documentation/English` directory.
2. Organize the material into a set of topics and subtopics corresponding to the categories that you want displayed in the Help Browser.
3. Add descriptive cell tags to the reference material to identify the cells that will be displayed under each category.
4. Edit the `BrowserCategories.m` file to create the categories that will appear in the Browser window. You should first save a copy of the existing `BrowserCategories.m` file in the same location under a different name, in case you need to revert to it later.
5. Edit the `BrowserIndex.nb` file to specify the keywords that will appear in the Help Browser's Master Index.

Specific details on each of these steps is provided in the rest of this article.

Note: Each time you modify any of the help files, you should click the `Help > Rebuild Help Index` menu command for your changes to take effect.

■ Setting Up New Browser Categories

□ Introduction

The four columns in the Help Browser display the categories into which the reference material is organized. These categories are arranged in a hierarchy with the first column showing the top-level categories. Clicking a topic either opens more subtopics in the next column or causes reference material about that topic to be displayed in the Browser window.

To make your own material accessible from the Help Browser, you need to define a set of topics that will be displayed in the Help Browser's columns. To do this, you must edit one of the `BrowserCategories.m` files contained in the `Documentation` directory of your *Mathematica* layout.

□ The `BrowserCategory` Command

The following example shows the contents of a simple `BrowserCategories.m` file.

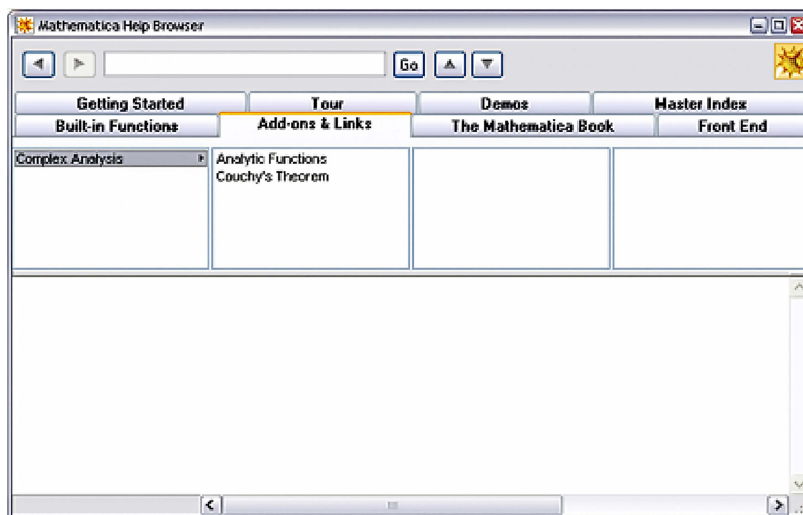
```

BrowserCategory["Add-ons", None,
  {BrowserCategory["Complex Analysis", None,
    {Item["Analytic Functions", "MyHelp.nb"],
     Item["Cauchy's Theorem", "MyHelp.nb"]}]}]
}]

```

The contents of a simple `BrowserCategories.m` file.

To investigate the effect of this file, copy it into the `Documentation/English/Add-Ons` directory of your *Mathematica* layout. You should first save a copy of the existing `BrowserCategories.m` file in the same location under a different name so that you can revert to it later. If you then rebuild the Help Browser (using the `Rebuild Help Index` command under the `Help` menu), the categories defined by the new `BrowserCategories.m` file will appear in the Browser window.



Every `BrowserCategories.m` file consists of a single `BrowserCategory` command of the form: `BrowserCategory["category name", name, list]`.

The arguments in a `BrowserCategory` command are as follows:

1. *"category name"*—This is a string that specifies the category name to be displayed in the Help Browser column.
2. *name*—If set to `None`, this option indicates that the source files are located in the same directory as the `BrowserCategory.m` file being edited. Alternatively, the source files can be kept in a subdirectory one level below the current directory. In this case *"name"* is set to the name of the subdirectory.
3. *list*—This is a list whose elements can be commands of the following types:
 - `Item["subcategory name", "file name", options]`—This command specifies the name of a terminal subcategory in the Help Browser.

Clicking the subcategory name causes information about it to be displayed in the Help Browser.

- `Item[Delimiter]`—This command inserts a horizontal line in the column where the subcategories are listed. It is useful for separating subcategories.
- `HelpDirectoryListing["directory", arg2, arg3]`—This command creates categories in the Help Browser's columns taken from a different `BrowserCategories.m` file. The location of the file to be used is specified by the three arguments.
- `BrowserCategory[]`—This command is of exactly the same form as the top-level `BrowserCategory` command, except that it is nested one level deeper in the hierarchy, so that its category name will be displayed one column to the right.

Note: Since there are only four columns in the Help Browser, a `BrowserCategory` command can be nested at most four levels deep.

□ The `HelpDirectoryListing` Command

The `HelpDirectoryListing` command is used to create categories in the Help Browser's columns based on a `BrowserCategories.m` file at a different location. It has the form: `HelpDirectoryListing[directory, arg2, arg3]`.

The three arguments *directory*, *arg2*, and *arg3* together specify the location of the `BrowserCategories.m` file to be used. The first argument, *directory*, specifies a directory location using the `FrontEnd'FileName` function. The absolute pathname of the directory must be given. Otherwise, the pathname specified is interpreted to be relative to the `Documentation/English` directory inside the top-level *Mathematica* directory.

If *arg2* is set to `False`, the Browser searches for a `BrowserCategories.m` file at the top level of "*directory*". If it is set to `True`, the Browser searches all subdirectories located one level below *directory*.

If *arg3* is set to `False`, the location of the `BrowserCategories.m` file is completely specified by the first two arguments. If it is set to `True`, the `BrowserCategories.m` file is taken to be in a `Documentation/English` directory located in the directory specified by the first two arguments.

If either of the last two arguments is omitted, they assume their default values. The default value of *arg2* is `True`. The default value of *arg3* is the current value of *arg2*.

These comments are summarized in the following table, which shows the directory that is searched to locate a `BrowserCategories.m` file for different values of *arg1* and *arg2*.

<i>arg2</i>	<i>arg3</i>	Location searched
False	False	<i>directory</i>
False	True	<i>directory</i> / Documentation / English
True	False	<i>directory</i> / *
True	True	<i>directory</i> / * / Documentation / English

Note: Here *directory* / * refers to all subdirectories located one level below *directory*.

Example 1

The `HelpDirectoryListing[]` command is used in the top-level `BrowserCategories.m` file in the `Documentation/English` directory to define the buttons that appear in the Help Browser window. Here are the contents of this file.

```
BrowserCategory["Help Browser", None, {
  HelpDirectoryListing[{"RefGuide"}, False],
  HelpDirectoryListing[{"AddOns"}, False],
  HelpDirectoryListing[{"MainBook"}, False],
  HelpDirectoryListing[{"OtherInformation"}, False],
  HelpDirectoryListing[{"GettingStarted"}, False],
  HelpDirectoryListing[{"Tour"}, False],
  HelpDirectoryListing[{"Demos"}, False],
  BrowserCategory["Master Index", None, {HelpMasterIndex[]}]
}]
```

Note that in each `HelpDirectoryListing` command here, *arg2* is omitted. This means it has the same value as *arg3* by default.

Example 2

Here is another simple example of a `BrowserCategories.m` file that uses `HelpDirectoryListing` commands. Here it is assumed that each directory, Chapter 2, Chapter 3, and so on, contains a separate `BrowserCategories.m` file.

```
BrowserCategory["Add-ons", None,
{ BrowserCategory["My Book", None, {
: BrowserCategory["Chapter 1", None,
{Item["Section 1.1", "Chapter1.nb"],
Item["Section 1.2", "Chapter1.nb"]}],
Item[Delimiter],
HelpDirectoryListing[{FrontEnd\FileName[{"AddOns",
"Chapter 2"}]}], False],
HelpDirectoryListing[{FrontEnd\FileName[{"AddOns",
"Chapter 3"}]}], False],
HelpDirectoryListing[{FrontEnd\FileName[{"AddOns",
"Chapter 4"}]}], False],
HelpDirectoryListing[{FrontEnd\FileName[{"AddOns",
"Chapter 5"}]}], False]
}]
```

□ The Item Command

The `Item` command has the form: `Item[subcategory name, file name, options]`.

The arguments of the `Item` command specify the target material that is displayed in the Help Browser when the subcategory name associated with that item is clicked.

1. *subcategory name*—This is a string that specifies the subcategory name to be displayed in the Help Browser column.
2. *file name*—This is a string that specifies the name of the notebook from which information pertaining to that subcategory is to be retrieved.
3. *options*—There can be two types of options: `IndexTag` and `CopyTag`:
 - The `IndexTag` serves as a tag to identify the item that is the target of a hyperlink in a notebook. A discussion of this option is deferred to the next section.
 - The `CopyTag` specifies which cells, from the notebook specified by *file name*, appear in the Help Browser when the item is clicked.

The value of `CopyTag` serves as a marker identifying the target cells linked to a specific topic. When an item in the Help Browser is clicked the Browser looks for cells, in the notebook specified by "*file name*", whose cell tags match the value of the `CopyTag` for that item. All such cells are then displayed in the Browser window.

The CopyTag option can be specified in four ways:

1. If `CopyTag` $\rightarrow$ `None`, the entire notebook specified by *file name* is displayed in the Browser window.
2. If `CopyTag` $\rightarrow$ "`tag`", the Browser looks for all cells in the notebook specified by *file name* whose cell tag is "`tag`". Only those cells are then displayed in the Browser window.
3. If the `CopyTag` is not specified explicitly, it takes the value of the `IndexTag` option.
4. If both the `CopyTag` and the `IndexTag` are omitted, they assume a default value given by the *subcategory name* specified in the first argument of the `Item` command.

□ An Example with Nested Categories

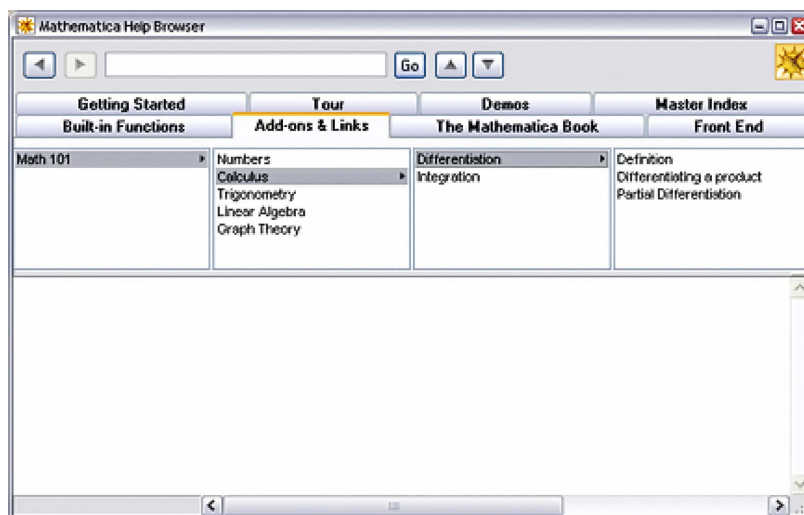
Here is a more complicated example illustrating how to define subcategories at different levels in the hierarchy. It contains multiple `BrowserCategory` commands, including one that is nested three levels deep. At each level of nesting, the category name associated with the `Item` command shifts one column to the right in the Browser window.

Note: The CopyTag is not specified explicitly for any of the Item commands in this example. It therefore assumes a default value given by the "subcategory name" for each item.

```
BrowserCategory["Add-ons", None,
  BrowserCategory["Math 101", None, {
    {Item["Numbers", "MyMathCourse.nb", IndexTag → "Chapter 1"],
    {BrowserCategory["Calculus", None,
      {BrowserCategory["Differentiation", None,
        {Item["Definition", "MyMathCourse.nb",
          IndexTag → "Subsection 2.1.1"],
          Item["Differentiating a product", "MyHelp.nb",
            IndexTag → "Subsection 2.1.2"],
          Item["Partial Differentiation", "MyHelp.nb",
            IndexTag → "Subsection 2.1.3"]}},
        Item["Integration", "MyMathCourse.nb", IndexTag →
          "Section 2.2"]}},
      Item["Trigonometry", "MyMathCourse.nb", IndexTag →
        "Chapter 3"],
      Item["Linear Algebra", "MyMathCourse.nb", IndexTag →
        "Chapter 4"],
      Item["Graph Theory", "MyMathCourse.nb", IndexTag →
        "Chapter 5"]}]
  }
}]
```

A sample BrowserCategories.m file showing nested commands.

Here are the categories defined by this file as they appear in the Browser window.



You can use the BrowserCategories.m file shown in this example as a template to construct your own Browser categories by replacing the chapters, sections, and

subsections in it with topics and subtopics relevant to your own material. You can add as many extra categories as you wish by adding more `BrowserCategory` or `Item` commands to the list at the level required.

Note: Since there are only four columns in the Help Browser, a `BrowserCategory` command can be nested at most four levels deep.

□ Adding a Sample Document

Here are the steps involved in setting up the Browser categories shown in the first example above.

1. Go to the Documentation/English directory in the top-level *Mathematica* directory and open the AddOns directory.
2. Rename the existing `BrowserCategories.m` file so that you can revert to it later. Then create another file called `BrowserCategories.m` containing the following text.

```
BrowserCategory["Add-ons", None,
{BrowserCategory["Complex Analysis", None,
{Item["Analytic Functions", "MyHelp.nb",
CopyTag→ "SubItem1"],
Item["Cauchy's Theorem", "MyHelp.nb",
CopyTag→ "SubItem2"]}]}]
```

3. Create a new notebook containing the following text.

Here is some reference material about analytic functions.
Here is some reference material about Cauchy's theorem.
4. Assign the cell tag "Complex Analysis" to the first cell and "Cauchy's theorem" to the second cell. To learn how to add a tag to a cell, see *Creating and Using Hyperlinks*.
5. Click the File ▸ **SaveAs** menu command to bring up the file dialog box.
6. Save this notebook as "MyHelp.nb" in the Documentation/English/Add-Ons directory.
7. Go to the Help menu and select **Rebuild Help Index**. This command updates the Help Browser so that it reflects your latest changes.

Note: You must add cell tags to any target material that is to be displayed when any item in the Browser window is clicked. The cell tags should match the `CopyTag` for that item, as explained in the discussion of the `Item` command.

You are now ready to view your help information.

1. Click Help ▸ Help Browser.
2. Click the Add-ons & Links button.

3. Click "Complex Analysis" in the leftmost category column and then click "Analytic Function" in the second column.

The first cell of your help notebook is displayed in the Browser window. If you click "Cauchy's theorem", the second cell will be displayed.

■ Adding Entries to the Master Index

□ Creating a New Index Entry

Once you have defined Browser categories for your reference material, you might want to incorporate keywords from it into the Master Index. The Master Index provides an alphabetical listing of all keywords, allowing users to quickly locate information on a specific topic.

To add new keywords to the Master Index:

1. Edit the `BrowserIndex.nb` file to add new entries and links.
2. Edit the `BrowserCategories.m` file to define `IndexTag` for each `Item` command.

We first illustrate this process with an example and then give details about the syntax of the `BrowserIndex.nb` file.

□ The `BrowserIndex.nb` file

The `BrowserIndex.nb` file is a *Mathematica* notebook that contains the list of terms that appear in the Master Index, as well as links to information about each item. This file must use the "HelpBrowser.nb" style sheet, which contains definitions for all the special styles needed. Here is an example of a simple `BrowserIndex.nb` file.

□ My Help Documentation

Numbers, 1

Calculus, 2

derivatives, 2.1

integrals, 2.2

Algebra, 3, A1.1

The contents of a simple `BrowserIndex.nb` file.

Every `BrowserIndex.nb` file consists of a Header followed by a list of entries. The first cell in the file is called the Header.

The Header must have the following properties:

- It must be a cell of style `IndexSection`.
- It must have `MasterIndexHeading` as a cell tag.

This is what the expression for the Header in the sample `BrowserIndex.nb` file looks like. You can see that it satisfies both of the previously listed properties.

```
Cell["My Help Documentation", "IndexSection",  
CellTags → "MasterIndexHeading"]
```

The Header cell is followed by a series of entries. Each entry corresponds to a single topic in the Master Index.

An entry has the following properties:

1. It is a cell of `Index` style.
2. It contains two items:
 - Entry name—This is the text that appears in the Master Index heading. It serves as a label for the information that the link points to.
 - Link—This is a special type of hyperlink that brings up reference material corresponding to the entry name and displays it in the Help Browser.
3. It must have a cell tag that is the same as the entry name.

Here is the cell expression for the first entry in the sample `BrowserIndex.nb` file. You can see that it has all the three properties listed earlier.

```
Cell[TextData[{"My first entry, ", ButtonBox["1",  
ButtonData:> "Chapter 1", ButtonStyle → "AddOnsLink"]}],  
"Index", CellTags → "My first entry"]
```

■ Creating Hyperlinks to the Help Browser

□ Creating a Button Box

The `CopyTag` option to the `Item` command allows you to specify which material should show up in the Help Browser when you click on a topic displayed in one of the columns. However, the `Item` command can also take a second type of option called `IndexTag`. Using the latter option, you can construct special hyperlinks in a notebook that point to reference material in the Help Browser.

When one of these hyperlinks is clicked, the Help Browser searches for the `Item` command in the `BrowserCategories.m` file that contains the corresponding `IndexTag`. The cells associated with that item are then displayed in the Help Browser.

To construct a hyperlink that uses an index tag, you should know how to unformat a cell and edit the expression representing the cell directly. To see the expression for a cell, select the cell and choose `Format ▸ Show Expression`. The expression for each hyperlink consists of a `ButtonBox` command with the following syntax:

```

ButtonBox["Button text", ButtonData:> "IndexTag",
ButtonStyle → "AddOnsLink"]

```

A `ButtonBox` command can have three arguments:

1. "Button text"—This is the text that forms the hyperlink. It usually appears underlined and in a different color.
2. "ButtonData"—This is an option that is used to specify the material referred to by the hyperlink. It is usually set to the `IndexTag` associated with the cell or group of cells that must be displayed in the Help Browser. If this option is omitted, then it assumes a default value given by "Button text".
3. "ButtonStyle"—This option specifies the directory in which the target material of the hyperlink is stored. It can take any of five values, each named after one of the button categories, that is, `GettingStartedLink`, `MainBookLink`, `OtherInformationLink`, `RefGuideLink`, or `AddOnsLink`.

This `ButtonData` option can be set in three different ways:

- i. If `ButtonData → "tag"`, then the Browser searches the `Browser-Categories.m` file for an item with that `IndexTag`. The cells associated with that item (i.e., whose cell tags match the `CopyTag` or `IndexTag` for the corresponding `Item` command) are then displayed in the Browser window.
- ii. If `ButtonData → {"tag1","tag2"}`, then the Browser searches the `BrowserCategories.m` file for an item with an `IndexTag` of "tag1". All cells associated with that item are then displayed in the Browser window. However, the window scrolls down to the first cell with the additional cell tag of "tag2".
- iii. If you do not set the `ButtonData` option explicitly, it assumes a default value given by the "subcategory name" specified in the first argument of the `Item` command.

An Example

Suppose you want to construct a hyperlink that points to information about the `Option Inspector` menu command under the `Format` menu. The cells with this information are located in the a notebook called "FormatMenu.nb" and have the cell tag "OptionsDialog". This notebook is in the `OtherInformation` directory of the `Documentation/English` directory. Here is the item in the `Browser-Categories.m` file that refers to this topic.

```

Item["Option Inspector","FormatMenu.nb", CopyTag →
"OptionsDialog", IndexTag → "Option Inspector"]

```

Hence the expression for the `ButtonBox` command will be the following.

```
ButtonBox["Option Inspector", ButtonData:> "Option Inspector",
ButtonStyle → "OtherInformationLink"]
```

Note

1. The first argument of the `ButtonBox` command contains the text that will represent the hyperlink.
2. The setting for the `ButtonData` option matches the `IndexTag` for this item.
3. The setting for the `ButtonStyle` option is `OtherInformationLink` because the target material is located in the `OtherInformation` subdirectory.

Here are the steps involved in creating such a hyperlink:

1. Select the cell in which you want to place the hyperlink, for example, a blank text cell.
2. Unformat the expression for the cell by typing `⌘⇧E`. (For Macintosh, type `⌘⇧E`.)
3. Insert the previous `ButtonBox` expression into the cell.
4. Wrap the command `TextData[ ]` around the `ButtonBox` expression inside the cell.
5. The cell expression now looks like this.

```
Cell[TextData[ButtonBox["Option Inspector", ButtonData
:> "Option Inspector", ButtonStyle → "Hyperlink"]],
"Text"]
```

6. Format the cell expression by typing `⌘⇧E` again.

This completes the construction of the hyperlink. The text "Option Inspector" now appears underlined and colored. Clicking the link will cause the Help Browser to open and display information about this menu command. A similar procedure can be followed for constructing a hyperlink to any other item in the Help Browser.

About the Author

Pavi Sandhu has a Ph.D. in physics from Boston University. He is the author of *The MathML Handbook*, published by Charles River Media in November 2002.

Pavi Sandhu

Senior Technical Writer
Oracle Corporation
Redwood Shores, CA

The Second Set of Magic Angles of Projectile Motion

Haiduke Sarafian

By suppressing the kinematics of projectile motion, we show that a projectile thrown at 49.50° above the horizontal maximizes the trajectory's perimeter. We also identify two unique projectile angles, 15.5° and 88.9° , that give two different parabolas with identical perimeters and identical areas.

■ The Magic Angles of Projectile Motion

A projectile thrown from the ground in a vacuum with an initial speed v_0 at an angle θ above the horizontal has a trajectory of length $L(v_0, \theta)$ and encompasses an area $\text{Area}(v_0, \theta)$ [1].

$$L(v_0, \theta) = \frac{v_0^2}{g} \left\{ \sin(\theta) - \cos^2(\theta) \ln \left[\frac{\cos(\theta)}{1 + \sin(\theta)} \right] \right\},$$

$$\text{Area}(v_0, \theta) = \frac{2}{3} \left(\frac{v_0^2}{g} \right)^2 \cos(\theta) \sin^3(\theta).$$

The length can be rewritten using $\text{gd}^{-1}(\theta) = -\ln \left[\frac{\cos(\theta)}{1 + \sin(\theta)} \right]$, an inverse Gudermanian function. In addition, a projectile has range

$$R(v_0, \theta) = \frac{v_0^2}{g} \sin(2\theta).$$

These equations share a peculiar property: the kinematics of the problem, the length $\frac{v_0^2}{g}$, is a factor separate from the parts that depend on the angle. This separation comes from the parametric representation of the projectile in Cartesian coordinates,

$$\{x(t), y(t)\} = \left\{ v_0 \cos(\theta) t, -\frac{1}{2} g t^2 + v_0 \sin(\theta) t \right\}. \quad (1)$$

Setting the kinematic factor $K = \frac{v_0^2}{g}$ to 1 allows us to study the global features of the purely geometric aspects of projectile motion.

The given length, area, and range for $K = 1$ are

```

In[1]:= length[θ_] := Sin[θ] - Cos[θ]^2 Log[ $\frac{\text{Cos}[\theta]}{1 + \text{Sin}[\theta]}$ ]
        area[θ_] :=  $\frac{2}{3}$  Cos[θ] Sin[θ]^3
        range[θ_] := Sin[2 θ]

```

Here is the trajectory's perimeter.

```

In[4]:= perimeter[θ_] := length[θ] + range[θ]

In[5]:= Plot[{perimeter[ $\frac{\pi}{180} t$ ], area[ $\frac{\pi}{180} t$ ]}, {t, 0, 90},
  PlotStyle -> {{}, {Dashing[{0.01}]}}, Frame -> True,
  FrameLabel -> {"θ", "perimeter, area", None, None},
  PlotRange -> {{0, 90}, {0, 2.2}}, GridLines -> Automatic]

```

From In[5]:=

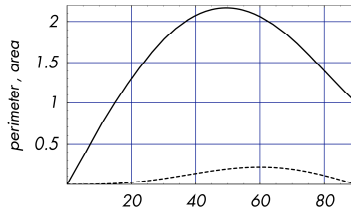


Figure 1. The perimeter (solid line) and the area (dashed line) versus the initial angle θ .

Contrary to our intuition, the longest perimeter does not encompass the largest area; 60° is the angle that maximizes the area.

```

In[6]:= Select[ $\frac{180}{\pi} \theta /. \text{Solve}[\text{D}[\text{area}[\theta], \theta] == 0, \theta], 0 < \# < 90 \&]$ 

```

Solve::ifun :
Inverse functions are being used by Solve, so some solutions may not
be found; use Reduce for complete solution information. [More...](#)

Out[6]= {60}

Similarly, to determine the angle (in degrees) that maximizes the perimeter, we set the slope of the perimeter to zero and solve the transcendental equation.

```

In[7]:=  $\frac{180}{\pi} \theta /. \text{FindRoot}[\text{D}[\text{perimeter}[\theta], \theta] == 0, \{\theta, 0, 1\}]$ 

```

Out[7]= 49.5097

```

In[8]:= ParametricPlot[{area[θ], perimeter[θ]}, {θ, 0,  $\frac{\pi}{2}$ },
  Frame → True, FrameLabel → {"area", "perimeter", None, None},
  PlotRange → {{0, 0.23}, {0, 2.3}}, GridLines → Automatic]

```

From In[8]:=

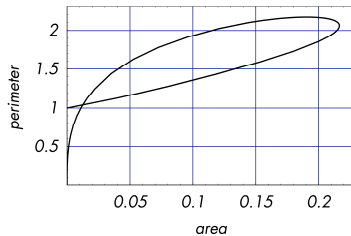


Figure 2. The perimeter versus the area for projectiles of initial angles $0 \leq \theta \leq 90^\circ$.

The point where the curve crosses itself determines two angles at which the corresponding pairs of areas and perimeters are the same. To evaluate these *magic* angles, we solve the simultaneous equations.

```

In[9]:= {α, β} =  $\frac{180}{\pi}$  {θ1, θ2} /.
  FindRoot[{area[θ1] == area[θ2], perimeter[θ1] == perimeter[θ2]},
    {{θ1, 0}, {θ2, .9 Pi / 2}},
    WorkingPrecision → 1.01 MachinePrecision] // Chop
Out[9]= {15.52988588878259, 88.9398614918464}

```

To plot the unique pair of corresponding kinematic-independent parabolic trajectories, we eliminate t between $\{x(t), y(t)\}$ as given in equation (1).

```

In[10]:= y[θ_, x_] :=
  y /. Solve[Eliminate[{x == v0 Cos[θ] t, y == - $\frac{g t^2}{2}$  + v0 Sin[θ] t,
     $\frac{v_0^2}{g} == 1$ }, {t, v0, g}], y][[1]]

```

```

In[11]:= y[θ, x]

```

```

Out[11]=  $-\frac{1}{2} \sec^2[\theta] (x^2 - 2 x \cos[\theta] \sin[\theta])$ 

```

```

In[12]:= Needs["Graphics`FilledPlot`"];

```

```

In[13]:= FilledPlot[
  {y[α Degree, x], y[β Degree, x]}, {x, 0, range[α Degree]},
  Fills → {{1, Axis}, GrayLevel[0.7]}, {{2, Axis}, GrayLevel[0.5]}},
  Curves → Front, Frame → True, FrameLabel → {"x", "y", None, None},
  PlotRange → {{0, 0.52}, {0, 0.52}}, GridLines → Automatic]

```

From In[13]:=

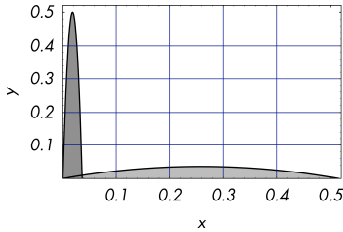


Figure 3. The magic pair of kinematic-independent trajectories. These two parabolas have the same perimeters and the same areas.

■ Acknowledgment

The author would like to thank the referee's useful suggestions.

■ References

- [1] H. Sarafian, "The Magic Angles of Projectile Motion," *Mathematica in Education and Research*, **9** (3-4), 2000 pp. 20–26.

About the Author

Haiduke Sarafian is professor of physics at the York campus of the Commonwealth College of The Pennsylvania State University. He received his Ph.D. in theoretical nuclear physics from Michigan State University in 1983. He has also been a research associate at the National Superconducting Cyclotron Laboratory in Michigan and a visiting faculty member at Tokyo Metropolitan University in Japan and the University of Valencia in Spain. In 1999, he received a *Mathematica* Visiting Scholar Grant and is an independent *Mathematica* trainer (www.wolfram.com/services/training/sarafian.html).

Haiduke Sarafian
 Commonwealth College
 The Pennsylvania State University
 Department of Physics
 1031 Edgecomb Avenue
 York, PA 17403
has2@psu.edu

Spatial Inversion: Reflective Anamorphograms

Philip W. Kuchel

The British flag (or 'Union Jack') is transformed so that if it is viewed from above with a straight-sided reflective cone placed in its center, the viewer perceives its original form to lie in the plane at the base of the cone. This so-called anamorphic art is readily studied by using the versatile graphics functions of *Mathematica*.

■ Introduction: Inversion of Space and Optical Transforms

This article discusses the fascinating idea that you can take the entire region of a two-dimensional surface between the outer edge of a circle and infinity (in all directions) and compress it into the interior of the circle with a single mathematical transformation. *Circle inversion* is one way to do this, and an explanation of its properties, when applied to simple geometrical objects, constitutes an important branch of geometry [1]. However, classical planar circle inversion is merely a special case of more general inversion transformations that can be applied to geometrical objects around a circle. In turn, these planar transformations can be generalized to three (and more) dimensions. Inversion in a sphere can be usefully employed to generate new curvilinear coordinate systems from older or more familiar ones [2–5]. If the coordinate lines intersect at right angles, the transformed ones will also be orthogonal because inversions preserve angles. Some rather extraordinary coordinate systems and corresponding surfaces can be generated in this way [2, 3].

The curved mirrors in fun houses elicit reactions of delight through the unexpected transformations they bring about when forming the images of the beholder. Likewise, the direction signs on road surfaces have strange proportions when viewed directly from above, and yet when viewed obliquely from the driver's position in a car, they appear perfectly well proportioned. These are but two examples of what is generally called *anamorphic* art from the Greek (ana = again, morphe = form) [6]. The transformation in the first case is wrought by the wavy mirror and can be classified as a reflective transformation: the undistorted object is in front and the image is perceived by the viewer to be behind the mirror. In the second case, the transformation of the object is achieved by changing the

angle of its projection onto the imaging surface that in most cases is the retina of the eye, or the film (or charge-couple device) of a camera.

■ Circle Inversion

Circle inversion entails taking a point O in the plane and specifying it as the center of a circle C of radius a . A point P is transformed to the point P' such that [1]

$$\overline{OP} \cdot \overline{OP'} = a^2. \quad (1)$$

It is relatively simple to prove a range of theorems involving this transformation.

1. A circle that is completely outside C is transformed to a circle wholly inside C , but not passing through the O (the center of inversion), and vice versa.
2. A circle that intersects C and passes through O is transformed to a straight line that passes through the points of intersection of the two circles, and vice versa.
3. The inverse of a line that does not pass through C is a circle inside C that passes through O , and so forth [1].

■ Reflective Circle Inversion

Circle inversion can also be invoked optically by using a conoidal reflecting surface that was described several years ago [7] (Figure 1). The object lies in the plane outside the base of the conoid. When the conoid is viewed directly from above its apex, an image appears to lie in the plane within the confines of its base. The aforementioned theorems can be readily verified with this device. Since the transformation is so regular, the transforms of various objects or anamorphograms can be made by using a ruler and compass. Specifically, straight lines in the image are made from arcs of circles that are drawn with the compass in the object field. However, this geometrical construction is tedious, and it is essentially redundant now that it can be performed so readily using *Mathematica*.

In the next section, we show how to take the reflective inversion of the Union Jack in a very direct manner. In order that the reflective surface can be easily constructed, rather than having to turn a circle inverting conoid from metal on a lathe [7], we have used the transformation that describes the reflection invoked by a straight-sided cone. Such cones can be made from discs of shim brass or aluminum-coated Mylar sheeting, or turned from solid metal if a lathe is accessible.

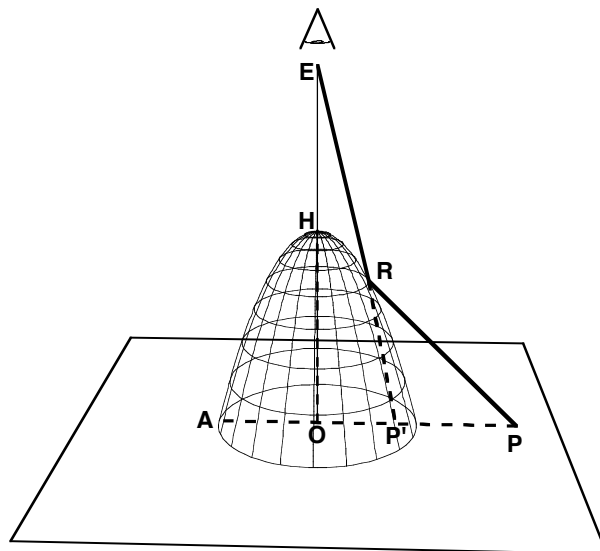


Figure 1. The arrangement of a circle-inverting reflective conoidal anamorphoscope [7]. The object field is a Euclidean plane that contains the anamorphogram, or picture, which is transformed into a familiar image that is perceived to lie within the confines of the base of the conoid (image field) when it sits on the plane and is viewed directly from its apex. The height of the conoid is h , the radius of the base is a , and the inverse of the point P is P' , such that it conforms to equation (1). The angle of incidence of the light from P' at the surface of the conoid R is equal to its angle of reflection up to the eye at E .

■ Reflective Inversion in a Simple Cone

Suppose that we use the general setup shown in Figure 1, but instead employ a straight-sided cone [7]. The path of a ray of light from the object field to the eye is indicated by the dashed line, while the perceived origin of the light is at the point of intersection of the thinner dashed line and the base plane (point P). For simplicity, it is assumed that the point of observation is at infinity above the plane of the object; hence, the light ray is specified as being parallel to the axis of symmetry of the cone. This is a mild approximation to having a viewing height of ≥ 20 times the height of the cone. The transformation rule is derived by considering Figure 2; here are the only physical rules to be used in arriving at the transformation formulas.

1. The angle of incidence of the light ray at the surface of the cone is equal to the angle of reflection.
2. The perceived source of a light ray lies at the intersection of the straight line drawn from the eye, intersecting the cone at the point of reflection and then intersecting the base plane.

3. When the solid angle at the vertex of the cone is 90° , the light from infinity is perceived to originate at the center of inversion. If the solid angle is less than 90° , then the horizon circle of the object field is not at infinity.

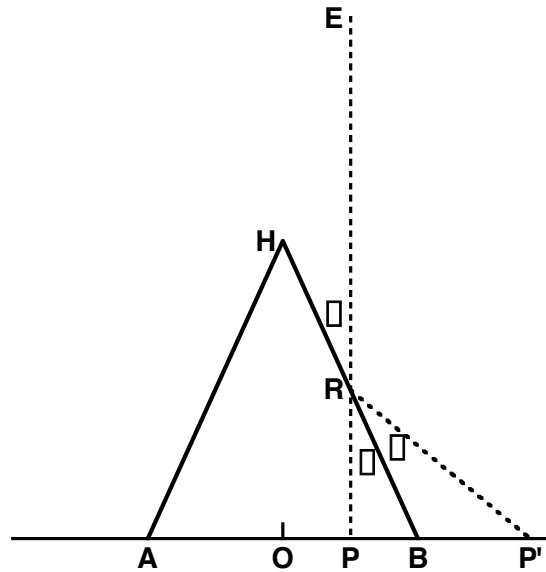


Figure 2. The central longitudinal section of a straight-sided cone showing the construction lines and light rays that are necessary to develop the mathematical expressions for the reflective inversion that is generated by such a cone. The symbols on the diagram correspond to those in Figure 1. The mathematical expressions are given in equations (2) to (5) for which $\alpha = \overline{OB}$ and $h = \overline{OH}$. In addition, α is the angle of incidence and reflection of the light ray, while, in general, for a straight-sided cone $\overline{OP} \cdot \overline{OP'} \neq \alpha^2$.

Therefore, to construct an anamorphogram, we begin with the image that will be produced. It is composed of line segments joining (x, y) coordinate points. The transformation of these to give the new points, in what will be the object field, is given after some trigonometric analysis by

$$x \rightarrow x' = \frac{x(d_1(x^2 + y^2)^{1/2} + d_2)}{(x^2 + y^2)^{1/2}} \quad (2)$$

$$y \rightarrow y' = y \frac{x'}{x}, \quad (3)$$

where the so-called dilation factors are

$$d_1 = 1 - \frac{h}{r} \quad (4)$$

$$d_2 = h t, \quad (5)$$

where h is the height of the cone, r is the radius of its base, and

$$t = \text{Tan}[2 \text{ArcTan}[r/b]]. \quad (6)$$

Let us now consider a familiar emblem that has a particularly intriguing transform.

■ Inverting the Union Jack

The British flag (or Union Jack) is especially striking because of its numerous angles, bright colors, and yet geometrically simple construction. First, the flag is defined geometrically by representing it as a set of triangles, parallelograms, and more general polygons, in which the sides are further divided into short linear segments. After the points are transformed, the sides of the resulting polygonal sections will be good approximations to smooth curves. Figure 3 shows such a representation of the Union Jack.

The Union Jack is not completely symmetric: it is intended to have a well-recognized top and bottom edge. In times of military or civilian emergency, it can be flown upside down to attract attention. Nevertheless it has two-fold rotational symmetry about its center so only two Cartesian quadrants need be represented as a list of coordinates, and the other two can be obtained by the appropriate rotation. Specifically, the upper-right quadrant is rotated through 180° to give the lower-left one, and the lower-right quadrant is rotated through 180° to give the upper-left one. The transformation that delivers this rotation through the angle α centered about the origin $(0, 0)$ is

$$x \rightarrow x' = x \text{Cos}[\alpha] + y \text{Sin}[\alpha] \quad (7)$$

$$y \rightarrow y' = x \text{Sin}[\alpha] + y \text{Cos}[\alpha], \quad (8)$$

and of course in the present case $\alpha = 180^\circ$.

The actual function used in the *Mathematica* program is called `rotator` and is implemented as follows.

```
rotator[points_] := Map[
  {{Cos[π], Sin[π], -Sin[π], Cos[π]}.# &,
  points];
```

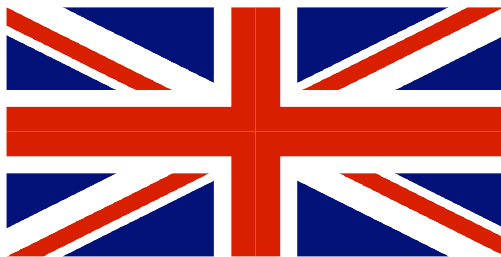


Figure 3. The Union Jack.

The inverted flag is produced by subjecting the List of coordinates to the transformations expressed in equations (2) to (5).

```

inverter[points_, h_, r_] := With[{
  t = Tan[2 ArcTan[ $\frac{r}{h}$ ]]},
  Map[{#[[1]], #[[2]] / #[[1]]} ((1 -  $\frac{h t}{r}$ ) Norm[#] + h t) / Norm[#] &,
  points]];

```

The inverted flag can be optically transformed back to the undistorted image of the Union Jack by placing a reflecting cone of the right height and radius at its center and viewing the whole setup from above the apex of the cone. Thus, we will have inverted the Union Jack mathematically and reinverted it optically (Figure 4).

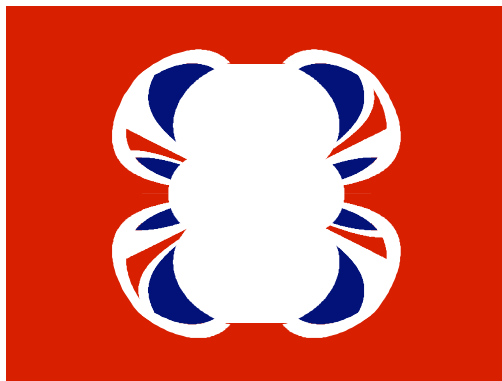


Figure 4. The reflective inversion of the Union Jack obtained by applying the coordinate transformation given by equations (2) to (5) to the polygonal (x, y) coordinates that define Figure 3. This figure is inverted back to an undistorted Union Jack by a straight-sided cone that has a base circle that touches the inside of the central four cusps of the pattern and is of a height that is $8/7$ of the diameter of the base; that is, the solid angle of the cone is 47.3° .

■ Conclusions

Mathematica provides an elegant way to set up geometrical objects and invoke reflective inversion transformations on them. In principle there is no limit to the implementation of reflective inversions of any reflective surface, even those that are not rotationally symmetrical.

The *Mathematica* notebook PolyFlag.nb performs the Union Jack transformation and the notebook DialInv.nb provides the more complicated example of inverting a clock face.

■ Acknowledgments

This paper was written while on a stimulating sabbatical leave with Professor Peter Stilbs at the Royal Institute of Technology-KTH in Stockholm. I would also like to thank Dr. Konstantin Momot for creating Figure 1 in *Mathematica*.

■ References

- [1] H. S. M. Coxeter, *Geometry*, Cambridge: Cambridge University Press, 1966.
- [2] P. Moon and D. Spencer, *Field Theory for Engineers*, Princeton: Van Nostrand, 1961.
- [3] P. Moon and D. Spencer, *Field Theory Handbook*, 2nd ed., Berlin: Springer-Verlag, 1967.
- [4] P. W. Kuchel, B. T. Bulliman, and E. D. Fackerell, "Bi-cyclide and Flat-ring Cyclide Coordinate Surfaces: Correction of Two Expressions," *Mathematics of Computation*, **49**, 1987 pp. 607–613.
- [5] P. W. Kuchel and B. T. Bulliman, "Expressions for Surfaces in Disk-cyclide Coordinates: Derivations using Symbolic Computation," *J. Franklin Inst.* **325**, 1988 pp. 505–508.
- [6] M. Gardner, "The Curious Magic of Anamorphic Art," *Scientific American*, **232**, 1975 pp. 110–116.
- [7] P. W. Kuchel, "Anamorphoscopes: A Visual Aid for Circle Inversion," *Math Gazette*, **63**, 1979 pp. 82–89.

■ Additional Material

PolyFlag.nb

DialInv.nb

Available at www.mathematica-journal.com.

About the Author

Philip W. Kuchel is McCaughey Professor of biochemistry at the University of Sydney, where he has been since 1980. He works in the general area of physical biochemistry and makes extensive use of nuclear magnetic resonance (NMR) spectroscopy to study biochemical reactions in cellular systems. His recent book, co-authored with Peter Mulquiney (*Modelling Metabolism with Mathematica*, (2003) CRC Press, Boca Raton), is an example of the extensive use of *Mathematica* in his research.

Philip W. Kuchel

School of Molecular and Microbial Biosciences,
University of Sydney, NSW 2006, Australia
p.kuchel@mmb.usyd.edu.au

Path Integral Methods for Parabolic Partial Differential Equations

With Examples from Computational Finance

Andrew Lyasoff

By using specific examples—mostly from computational finance—this paper shows that *Mathematica* is capable of transforming Feynman-type integrals on the pathspace into powerful numerical procedures for solving general partial differential equations (PDEs) of parabolic type, with boundary conditions prescribed at some fixed or free time boundary. Compared to the widely used finite difference methods, such procedures are more universal, are considerably easier to implement, and do not require any manipulation of the PDE. An economist or a financial engineer who works with very general models for asset values, including the so-called stochastic volatility models, will find that this method is just as easy to work with as the classical Black–Scholes model, in which asset values can follow only a geometric Brownian motion process.

■ 1. Introduction

Parabolic partial differential equations (PDEs), of which the standard heat equation is a special case, are instrumental for many applications in physics, engineering, economics, finance, weather forecasting, and many other fields. In practice, the solution to most parabolic PDEs can be computed only approximately by way of some special numerical procedure. Currently, in most if not all practical implementations, this procedure happens to be some variation of the well-known *finite difference method*.

This method involves tools and terminology that are familiar to most specialists in numerical methods. Its underlying principle is relatively easy to understand: it comes down to solving sequentially a very large number of linear systems.

Unfortunately, in most applications the size of each system is overwhelming and the practical implementation of the method is not straightforward, especially in the case of non-constant coefficients.

It has been well known for nearly as long as the finite difference method has been in existence that the solution of the heat equation can be expressed as a path integral (also known as a Feynman integral). For a mathematician, this is nothing but an expected value relative to Wiener's probability distribution on the space of sample paths. This construction also admits a generalization: the solution of a fairly general parabolic PDE can be expressed as an expected value (alias: a Feynman integral) relative to a special probability distribution on the pathspace (alias: the Wiener measure). Although possible in principle, the numerical calculation of path integrals has always been seen as a formidable task and a rather inefficient way to produce a solution to a parabolic PDE. As the examples included in this article demonstrate, not only is this no longer the case, but, with the advent of *Mathematica* 4, the solution to a rather general parabolic PDE can be produced by merely expressing the associated path integral in *Mathematica*.

The method developed in this paper is a good illustration of the way in which significant new developments in the common man's computing tools transform the type of mathematics that is commonly perceived as interesting and important. We will show with concrete examples (see Section 4 below) that computing the price of an European call option—a classical problem in computational finance—in terms of path integrals is just as easy, just as accurate, and virtually just as fast as calculating the price directly by using the Black–Scholes formula. In other words, for all practical purposes, the path integral representation of the solution of the Black–Scholes PDE in *Mathematica* is just as good as a closed form solution. Unlike the true closed form solution, however, the path integral representation can be used with any reasonably behaved payoff. Of course, this approach would have been unthinkable when Black and Scholes developed their method [1]. Their calculation was interesting and profoundly important, although the really important discovery that they made was the principle that led to the Black–Scholes PDE—not the Black–Scholes formula, as is commonly believed.

It is remarkable how *Mathematica*'s ability to combine advanced symbolic computing with advanced numerical procedures can turn a fairly abstract mathematical concept (in this case, integration on the pathspace) into a rather practical computing tool. This brings a method to consumers of computing technology that is both universal and “smart,” in the sense that it is capable of taking advantage of the structure of the PDE. For example, the method will need fewer iterations if the coefficients are close to linear, and can work on a very coarse grid in the state space if the solution is reasonably close to a polynomial. Above all, its implementation is extremely straightforward, involves very little programming, and does not require any knowledge of sophisticated numerical methods, which *Mathematica* uses implicitly. For a financial engineer, this means that pricing an option with any reasonably-behaved payoff function on an asset that follows a rather general time-homogeneous Markov process, including options that allow early exercise, would be just as easy as pricing a vanilla European call or put option on an asset

that follows a standard geometric Brownian motion process. Such technology can have a considerable impact on the financial industry. Indeed, it has long been recognized that actual stock prices exhibit self-regulatory patterns that are inconsistent with the properties of the geometric Brownian motion. Yet the Black–Scholes model remains immensely popular mainly because, with the current level of widely available computing technology, any other model would be virtually intractable for an average practitioner.

Finding the most efficient way to present the computational aspects of the path integral method and its implementation in *Mathematica* is a bit of a challenge. Indeed, on the one hand, specialists in (and consumers of) numerical methods for PDEs do not usually speak the language of path integrals. On the other hand, specialists in mathematical physics and probability, who do speak the language of path integrals, are not usually interested in numerical methods and computer programming. This article grew out of the desire to meet this challenge. In spite of its use of high-end mathematical jargon, it is mainly about *Mathematica* and the impact that *Mathematica* can make not only on the way in which important computational problems are being solved, but also on our understanding of which mathematical methods are “applied” and which are “theoretical.”

Note about the notation: Following the common convention in *Mathematica*, in all equations and algebraic expressions, arguments of functions are wrapped with square brackets and parenthesis are used only for grouping. Double-struck square brackets are used to denote intervals: open intervals are denoted as $\llbracket a, b \llbracket$ and closed intervals are denoted as $\llbracket a, b \rrbracket$.

For the purpose of illustration, all time-consuming operations in this notebook are followed by `//Timing`. The CPU times reflect the speed of a 2.8Ghz Intel® Pentium® 4 processor with 768MB of RAM running *Mathematica* 5.0.

■ 2. Some Background: Solutions to Second-order Parabolic Partial Differential Equations as Path Integrals

This section outlines some well-known principles and relations on which the method developed in this paper is based. A complete, detailed, and rigorous description of these principles can be found in the ground-breaking work of Stroock and Varadhan [2] and also in [3, 4]. The computational aspects of the methods developed in these works have not been explored fully, and to a large extent the present work is a modest attempt to fill this gap.

The *Feynman integral* of a real function $f[\cdot]$ defined on the space $C_0\llbracket t, T \rrbracket$, which consists of continuous sample paths $\omega : \llbracket t, T \rrbracket \mapsto \mathbb{R}$ with the property $\omega_t = 0$, can be expressed as the following formal expression

$$\int_{\{\omega \in C_0\llbracket t, T \rrbracket\}} f[\omega] e^{-\frac{1}{2} \int_t^T \left(\frac{d\omega_s}{ds}\right)^2 ds} \left(\prod_{s \in \llbracket t, T \rrbracket} \frac{d\omega_s}{\sqrt{2\pi}} \right), \quad (2.1)$$

which is understood as the limit of the following sequence of multiple integrals

$$(2\pi)^{-\frac{k}{2}} \int_{\mathbb{R}} \dots \int_{\mathbb{R}} f_k[\omega_0, \omega_d, \dots, \omega_{kd}] e^{-\sum_{i=1}^k \frac{(\omega_{id} - \omega_{(i-1)d})^2}{2d}} \times d(\omega_d - \omega_0) \dots d(\omega_{kd} - \omega_{(k-1)d}), \quad (2.2)$$

where $k \geq 1$, $d := \frac{T-t}{k}$ and $f_k[\omega_0, \omega_d, \dots, \omega_{kd}]$ is the result of the evaluation of $f[\cdot]$ at the piecewise linear sample path from the space $C_0[[t, T]]$, whose values at the points $t + id$, $0 \leq i \leq k$, are given by the vector $(\omega_0, \omega_d, \dots, \omega_{kd})$. It would not be an exaggeration to say that the convergence of the sequence in (2.2)—and therefore the fact that the integral in (2.1) is meaningful—is a miracle, without which very little of what we currently know about the universe would have taken place. Indeed, “miracle” is the only term that can explain the fact that, as was pointed out in [4], the infinities that arise under the exponent in (2.1) not only offset the infinities that arise under the product sign, but also offset them in such a way that the result is a perfectly well defined and highly nontrivial probability measure on the space $C_0[[t, T]]$, known as the *Wiener measure*. Not only is the integral in (2.1) the perfectly meaningful “expected value of $f[\cdot]$,” but when $f[\cdot]$ is given by $f[\omega] := \Lambda[x + \omega_T]$, $\omega \in C_0[[t, T]]$, for some fixed $x \in \mathbb{R}$ and some reasonably behaved function $\Lambda: \mathbb{R} \mapsto \mathbb{R}$, then, as a function of x and t , this expected value coincides with the solution of the standard heat equation

$$\partial_t u[t, x] + \mathcal{L} u[t, x] = 0, \quad \mathcal{L} := \frac{1}{2} \partial_{x,x}, \quad x \in \mathbb{R}, \quad t \leq T, \quad (2.3)$$

with boundary condition $u[T, x] \equiv \Lambda[x]$. In fact, this statement admits a generalization: if $\mathcal{L}$ is replaced by a general elliptic differential operator of the form

$$\mathcal{L} = \frac{1}{2} \sigma^2[x] \partial_{x,x} + a[x] \partial_x, \quad (2.4)$$

assuming that the coefficients $\sigma[\cdot]$ and $a[\cdot]$ are reasonably behaved, the solution of the boundary value problem (2.3) can still be expressed as a path integral similar to the one in (2.1). The only difference is that in this case the path integral is understood as the limit of the sequence

$$\int_{\mathbb{R}} d(\omega_d - x) p_d[x, \omega_d] \times \int_{\mathbb{R}} d(\omega_{2d} - \omega_d) p_d[\omega_d, \omega_{2d}] \times \dots \times \int_{\mathbb{R}} d(\omega_{kd} - \omega_{(k-1)d}) p_d[\omega_{(k-1)d}, \omega_{kd}] \Lambda[x + \omega_{kd}], \quad k \geq 1, \quad (2.5)$$

where $p_t[x, y]$ is the transition probability density of the Markov process X_t , $t \geq 0$, determined by the stochastic equation

$$dX_t = \sigma[X_t] d\mathfrak{B}_t + a[X_t] dt, \quad (2.6)$$

in which $\mathfrak{B}_t$, $t \geq 0$, is a given Brownian motion process. In other words, $(t, x, y) \mapsto p_t[x, y]$ is characterized by the fact that for any test function $\varphi[\cdot]$ one has

$$E[\varphi[X_{t+b}] | X_t = x] = \int_{\mathbb{R}} \varphi[y] p_b[x, y] dy.$$

We will say that the process X_t , $t \geq 0$, is a *sample path realization of the Markovian semigroup* $e^{t\mathcal{L}}$, $t \geq 0$, $\mathcal{L}$ being the differential operator from (2.4), and will refer to the transition density $(t, x, y) \rightarrow p_t[x, y]$ as *the integral kernel of the semigroup* $e^{t\mathcal{L}}$, $t \geq 0$. This integral kernel is also known as *the fundamental solution of the equation* $(\partial_t - \mathcal{L})u[t, x] = 0$, that is, a solution which satisfies the initial condition $u[0, x] = \delta_y[x]$, where $\delta_y[\cdot]$ stands for the usual Dirac delta function with mass concentrated at the point $y \in \mathcal{D}$. This means that, formally, $p_t[x, y]$ can be expressed as $p_t[x, y] = (e^{t\mathcal{L}} \delta_y)[x]$.

A thorough exploration of this generalization of the Feynman integral can be found in the seminal work of Stroock and Varadhan [2]. The terminology is somewhat different from the one used here, that is, instead of “path integrals,” it comes down to a study of probability distributions on the pathspace associated with equations similar to the one in (2.6). Such a framework is much more rigorous and powerful. However, our goal is to develop a numerical procedure that computes the solution of the PDE in (2.3) as an expected value, and in this respect Feynman’s formalism proves to be quite useful.

■ 3. Path Integrals as Approximation Procedures

The multiple integral in (2.5) is actually an iterated integral, which can be transcribed as a recursive rule. More specifically, if for some fixed $T \geq 0$, the function $(t, x) \rightarrow u[t, x]$ satisfies the equation

$$(\partial_t + \mathcal{L})u[t, x] = 0, \quad \mathcal{L} \equiv \frac{1}{2} \sigma^2[x] \partial_{x,x} + a[x] \partial_x,$$

for $t \leq T$ and for $x \in \mathbb{R}$, assuming that the map $x \rightarrow u[T, x]$ is given, one can compute the maps $x \rightarrow u[T - id, x]$ consecutively for $i = 1, 2, \dots$, according to the following procedure

$$\begin{aligned} u[T - id, x] &= \int_{\mathbb{R}} p_d[x, y] \times u[T - (i - 1)d, y] dy \\ &= E[u[T - (i - 1)d, X_d] \mid X_0 = x], \quad i = 1, 2, \dots \end{aligned} \quad (3.1)$$

As we will soon see, the implementation of this procedure in *Mathematica* is completely straightforward. Since the above identity holds for every choice of $d > 0$, this raises the question: why does one need the recursive procedure at all? After all, one can always take $d = T - t$ and then use the recursive rule in (3.1) only for $i = 1$, which comes down to what is known as the *Feynman–Kac formula*:

$$\begin{aligned} u[t, x] &= \int_{\mathbb{R}} p_{T-t}[x, y] \times u[T, y] dy \equiv E[u[T, X_{T-t}] \mid X_0 = x] \\ &\equiv E[\Lambda[X_{T-t}] \mid X_0 = x]. \end{aligned} \quad (3.2)$$

Unfortunately, the integral kernel $(t, x, y) \rightarrow p_{T-t}[x, y] \equiv (e^{(T-t)\mathcal{L}} \delta_y)[x]$ can only be expressed in a form that allows one to actually compute this last integral—for example, by way of some numerical procedure—for some special choice of the coefficients $\sigma[\cdot]$ and $a[\cdot]$. The idea, then, is to use the recursive rule

(3.1) with some “sufficiently small” $d > 0$ and hope that for such a choice of d , $p_d[x, y]$ can be replaced with some approximate kernel, which, on the one hand, makes the integral (the expected value) on the right side of (3.1) computable and, on the other hand, does not allow the error in the calculation of $u[T - id, x]$ caused by this approximation to grow larger than $ido(d)$. We will do what appears to be completely obvious and intuitive: replace $\sigma[\cdot]$ and $a[\cdot]$ with their first-order Taylor expansions about the point $x \in \mathbb{R}$. Intuitively, it should be clear that when the process X_t , $t \geq 0$ is governed by the stochastic equation in (2.6) over a short period of time, its behavior cannot be very different from the behavior of a process that starts from the same position $X_0 = x$ and is governed by the first-order Taylor approximation of the stochastic equation in (2.6) around the point $x \in \mathbb{R}$. The reason why one has to settle for a first-order approximation is that, in general, only when the coefficients $\sigma[\cdot]$ and $a[\cdot]$ in (2.6) are affine functions can one solve this equation explicitly, that is, write the solution as an explicit function of the driving Brownian motion. In general, the probability density of this explicit function is not something that is known to most computing systems and therefore its use in the computation of the integral on the right side of (3.1) is not straightforward. One way around this problem is to compute, once and for all, the transition probability density of a Markov process governed by a stochastic differential equation (SDE) with affine coefficients, and then add that function of seven variables (four parameters determined by the two affine coefficients $\sigma[\cdot]$ and $a[\cdot]$ plus the variables t , x , and y) to the list of standard functions in the system. In this exposition we will not go that far, however. Instead, we will develop a second level of approximation, in which the solution of a general SDE with affine coefficients will be replaced by a computable function of the starting point x , the time t , and the position of the driving Brownian motion at time t . After this second level of approximation, the computation of the integral on the right side of (3.1) comes down to computing a Gaussian integral, which, as we will soon see, is not only feasible, but is also extremely easy to implement. Note that when the coefficients $\sigma[\cdot]$ and $a[\cdot]$ are linear functions, this second level of approximation of the solution of the stochastic equation in (2.6) actually coincides with the exact solution and is therefore redundant.

Before we can implement the plan that we just outlined, we must address an entirely mundane question: how is each function $y \rightarrow u[T - id, y]$, $i \geq 0$, going to be represented in the computing system? The answer is that we will use polynomial interpolation; that is, in every iteration, the integral on the right side of (3.1) (with the approximate integral kernel) will be computed only for

$$x \in \mathcal{P} := \left\{ \Sigma + j \frac{b}{m}; -m \leq j \leq m \right\},$$

where Σ , b , and m are given parameters. Furthermore, in the actual computation of the integral, the map $y \rightarrow u[T - (i - 1)d, y]$ will be replaced by the map constructed by way of polynomial interpolation (and extrapolation) from the values

$$u \left[T - (i - 1)d, \Sigma + j \frac{b}{m} \right], \quad -m \leq j \leq m.$$

In other words, our strategy is to store each of the functions $u[t, \cdot]$, for $t = T - id$, $i = 1, 2, \dots$, as a list of $2m + 1$ real values and interpret $u[t, \cdot]$ as the result of polynomial interpolation from that list. This means that we will be constructing the solution $u[t, x]$ only for $x \in [\Sigma - b, \Sigma + b]$ and $t = T - id$, $i = 1, 2, \dots$, for some appropriate choice of $d > 0$. In most practical situations, one needs to construct $u[t, \cdot]$ only on some finite interval and only for a fixed $t < T$. To do this with our method, one must set $d = \frac{T-t}{k}$ for some sufficiently large $k \in \mathbb{Z}_+$, and then iterate (3.1) k times after choosing Σ and b so that the interval of interest is covered by the interval $[\Sigma - b, \Sigma + b]$; the number of interpolation nodes $(2m + 1)$ must be chosen so that the polynomial interpolation is reasonably accurate. This might appear as a daunting programming task, but, as we will soon see, in *Mathematica* it involves hardly any programming at all. The reason is that the integration routine can accept as an input the output from the interpolation routine—all in symbolic form. This illustrates perfectly the point made earlier: new developments in the common man's computing tools can change the type of mathematics that one can use in practice.

A good point can and should be made that approaching (from a purely computational point of view) a general parabolic PDE like the one in (2.3) in the way we did and with the type of mathematics that we have chosen, namely, integration on the path space, is more than just natural. Indeed, the connection between the distribution of the sample paths of the process X_t , $t \geq 0$, governed by the stochastic equation in (2.6) and the parabolic PDE (2.3) runs very deep (see [2, 3] for a detailed study of this relationship). The point is that the distribution on the space of sample paths associated with the SDE (2.6) encodes all the information about the elliptic differential operator $\mathcal{L}$ from (2.4) and, therefore, all the information about the function

$$u[t, x] = (e^{(T-t)\mathcal{L}}) \Lambda[x], \quad x \in \mathbb{R}, \quad t \leq T.$$

This is analogous to the claim that the integral curves of a vector field encode all the information about the vector field itself. Actually, this is much more than a mere analogy. To see this, notice that if $\mathcal{L}$ were a first-order differential operator (i.e., a vector field) in the open interval $\mathcal{D} \subset \mathbb{R}$, if $t \rightarrow X_t$ were an integral curve for $\mathcal{L}$ starting from $X_0 = x$, and if $(t, x) \rightarrow u[t, x]$ has the property $(\partial_t + \mathcal{L})u[t, x] = 0$, then one would have

$$u[T, X_{T-t}] - u[t, x] = \int_t^T \partial_s u[s, X_{s-t}] ds = \int_t^T 0 ds = 0. \quad (3.3)$$

In other words, the integral curves of the vector field $\mathcal{L}$ can be characterized as “curves along which any solution of $(\partial_t + \mathcal{L})u[t, x] = 0$ remains constant.” This is interesting as well as quite practical. Indeed, if $(\partial_t + \mathcal{L})u[t, x] = 0$ must be solved for $t \leq T$ and $x \in \mathcal{D}$ with boundary condition $u[T, x] = \Lambda[x]$, $x \in \mathcal{D}$, one can compute the solution $u[t, x]$ by first solving the ODE $\frac{d}{dt} X_t = \mathcal{L}[X_t]$ with initial data $X_0 = x$. Here we treat $\mathcal{L}$ as a “true” vector field (i.e., as a map of the form $\mathcal{L}: \mathcal{D} \mapsto \mathbb{R}$) and then by evaluating the function $\Lambda[\cdot]$ (assuming that it is given) at the point X_{T-t} . Thus, the computation of $u[t, x]$, for a given $x \in \mathcal{D}$ and $t < T$,

essentially comes down to constructing the integral curve $\llbracket 0, T - t \rrbracket \ni s \rightarrow X_s$ for the vector field $\mathcal{L}$ with initial position $X_0 = x$.

Of course, the remark just made can be found in any primer on PDEs and demonstrates how natural our approach to parabolic PDEs actually is. Indeed, the Feynman–Kac formula in (3.2) is essentially an analog of the relation in (3.3) in the case where $\mathcal{L}$ is a second-order differential operator of elliptic type. This amounts to the claim that the solution of the parabolic PDE $(\partial_t + \mathcal{L})u[t, x] = 0$ remains *constant on average* along the sample path of the process X_t , $t \geq 0$, which is exactly the sample path representation of the semigroup $e^{t\mathcal{L}}$, $t \geq 0$. Clearly, for any given starting point $X_0 = x$, the probability distribution of the sample path of the process X_t , $t \geq 0$, has the meaning of a “distributed integral curve” for the second-order differential operator $\mathcal{L}$. This interpretation of the Feynman–Kac formalism is due to Stroock [3] and has been known for quite some time. The only “novelty” here is our attempt to utilize its computing power. The recursive rule (3.1) is analogous to a numerical procedure for constructing an integral curve of a standard (first-order) vector field by solving numerically a first-order ordinary differential equation.

One should point out that the main concept on which all finite difference methods rest is quite natural, too. Indeed, the intrinsic meaning of a partial derivative is nothing but “a difference quotient of infinitesimally small increments.” However, such a concept is intrinsic in general and not necessarily intrinsic for the equation $(\partial_t + \mathcal{L})u[t, x] = 0$. This is very similar to encoding an image of a circle as a list of a few thousand black or white pixels. There are various methods that would allow one to compress such an image, but they all miss the main point: a circle is determined only by its radius and center point, and this is the only data that one actually needs to encode. It is not surprising that, in order to achieve a reasonable accuracy, essentially all reincarnations of the finite difference method require a very dense grid on the time scale as well as the state-space.

As was pointed out earlier, path integrals can be used just as easily in the case of free boundary value problems of the following type: given an interval $\mathcal{D} \subset \mathbb{R}$ and a number $T \in \mathbb{R}_+$, find a function $\llbracket -\infty, T \rrbracket \times \mathcal{D} \ni (t, x) \rightarrow u[t, x] \in \mathbb{R}$ and also a free time-boundary $\llbracket -\infty, T \rrbracket \ni t \rightarrow x^*[t] \in \mathcal{D}$ so that

- 1) $u[t, x] \geq \Lambda[x]$, for $x \in \mathcal{D}$;
 - 2) $u[t, x] = \Lambda[x]$, for $x \in \mathcal{D} \cap \llbracket -\infty, x^*[t] \rrbracket$;
 - 3) $(\partial_t + \mathcal{L})u[t, x] = 0$, for $x \in \mathcal{D} \cap \llbracket x^*[t], \infty \rrbracket$;
 - 4) $\lim_{x \searrow x^*[t]} u[t, x] = \Lambda[x^*[t]]$ and $(\partial_x u)[t, x^*[t]] = \Lambda'[x^*[t]]$;
 - 5) $u[T, x] = \Lambda[x]$ for $x \in \mathcal{D}$.
- (3.4)

This is an optimal stopping problem: at every moment $t \leq T$, one observes the value of the system process X_t , $t \geq 0$, which is governed by the stochastic equation in (2.6), and decides whether to collect the termination payoff $\Lambda[X_t]$ or take no action and wait for a better opportunity. The function $(t, x) \rightarrow u[t, x]$, which satisfies the conditions in (3.4), gives the price at time t ($t < T$) of the right to collect the payoff at some moment during the time period $\llbracket t, T \rrbracket$, assuming that at time t the observed value of the system process is $X_t = x$. Also, at time t ($t < T$), the set $\llbracket -\infty, x^*[t] \rrbracket \cap \mathcal{D}$ gives the range of values of the system process

for which it is optimal to collect the payoff immediately, while the set $\llbracket x^*[t], \infty \rrbracket \cap \mathcal{D}$ gives the range of values of the system process for which it is optimal to wait for a better opportunity. The valuation and the optimal exercise of an American stock option with payoff $\Lambda[X_t]$, X_t being the price of the underlying asset at time t , is a classical example of an optimal stopping problem. Of course, here we assume that the exercise region has the form $\llbracket -\infty, x^*[t] \rrbracket$. This assumption is always satisfied if $\Lambda[\cdot]$ is a decreasing function of the system process, provided that larger payoffs are preferable. As it turns out, the recursive procedure (3.1) can produce, in the limit, the solution $\llbracket -\infty, T \rrbracket \times \mathcal{D} \ni (t, x) \rightarrow u[t, x]$ of the above optimal stopping problem with the following trivial modification:

$$u[T - i d, x] = \text{Max} \left[\Lambda[x], \int_{\mathbb{R}} p_d[x, y] \times u[T - (i - 1) d, y] dy \right], \quad (3.5)$$

$$i = 1, 2, \dots$$

This is simply a rephrasing of Bellman's principle for optimality in discrete time, and the conditions in (3.4) are simply a special case of Bellman's equation.

We conclude this section with the remark that the above considerations can easily be adjusted to the case where the elliptic differential operator $\mathcal{L}$ contains a 0-order term, that is,

$$\mathcal{L} = \frac{1}{2} \sigma^2[x] \partial_{x,x} + a[x] \partial_x - r \quad (3.6)$$

where $r \in \mathbb{R}$ is a given parameter. The reason why such a generalization is actually trivial is that when $(t, x) \rightarrow u[t, x]$ satisfies the equation $(\partial_t + \mathcal{L}) u[t, x] = 0$ with $\mathcal{L} = \frac{1}{2} \sigma^2[x] \partial_{x,x} + a[x] \partial_x$, then the function

$$(t, x) \rightarrow e^{-(T-t)r} u[t, x]$$

satisfies the same equation, but with $\mathcal{L} = \frac{1}{2} \sigma^2[x] \partial_{x,x} + a[x] \partial_x - r$. In any case, if $\mathcal{L}$ is chosen as in (3.6) and not as in (2.4), the only change that one will have to make in the procedure outlined earlier in this section would be to replace the recursive procedure in (3.1) and (3.5), respectively, with

$$u[T - i d, x] = \int_{\mathbb{R}} e^{-r d} p_d[x, y] \times u[T - (i - 1) d, y] dy,$$

$$i = 1, 2, \dots,$$

and

$$u[T - i d, x] = \text{Max} \left[\Lambda[x], \int_{\mathbb{R}} e^{-r d} p_d[x, y] \times u[T - (i - 1) d, y] dy \right],$$

$$i = 1, 2, \dots$$

■ 4. The Black–Scholes Formula versus the Feynman–Kac Representation of the Solution of the Black–Scholes PDE

In the option pricing model developed by F. Black and M. Scholes [1] the asset price S is assumed to be governed by the stochastic equation

$$dS_t = \varsigma S_t d\mathbb{B}_t + \mu S_t dt, \quad t \geq 0, \quad (4.1)$$

in which $\varsigma > 0$ and $\mu > 0$ are given parameters. At time $t > 0$ the price of an European call option with payoff $\Lambda[S] = \text{Max}[S - K, 0]$, which expires at time $T > t$, can be identified with the expression $u[t, S_t]$, where $(t, x) \rightarrow u[t, x]$ is the solution of the PDE:

$$\begin{aligned} \partial_t u[t, x] + \frac{1}{2} \sigma^2 x^2 \partial_x^2 u[t, x] + rx \partial_x u[t, x] - ru[t, x] = 0, \\ t \leq T, \quad x > 0, \end{aligned} \quad (4.2)$$

with boundary condition $\lim_{t \nearrow T} u[t, x] = \Lambda[x]$, where $r > 0$ is a parameter representing the (constant) short-term interest rate and $\Lambda[\cdot]$ gives the termination payoff as a function of the price of the underlying asset. The fact that the option price must satisfy the PDE (4.2) may be seen as a consequence of Bellman's principle for optimality, but one has to keep in mind that the use of this principle in finance is somewhat subtle. Notice, for example, that the Feynman–Kac formula (3.2) links the PDE (4.2) not with the stochastic equation in (4.1), but, rather, with the stochastic equation

$$dX_t = \varsigma X_t d\mathbb{B}_t + r X_t dt, \quad t \geq 0. \quad (4.3)$$

Therefore, the solution of the Black–Scholes equation in (4.2) can be expressed as an expected value of the form

$$u[t, x] = E[e^{-r(T-t)} \text{Max}[X_{T-t} - K, 0] \mid X_0 = x], \quad t < T, \quad x \in \mathbb{R}_+,$$

and *not* as

$$u[t, x] = E[e^{-r(T-t)} \text{Max}[S_{T-t} - K, 0] \mid S_0 = x], \quad t < T, \quad x \in \mathbb{R}_+.$$

In other words, in the context of option pricing, one must use Bellman's principle as if the predictable part in the system process has the form $r X_t dt$, regardless of the form that the predictable part in the system process actually has. This phenomenon is well known and its justification, of which there are several, can be found in virtually any primer on financial mathematics. Roughly speaking, this is caused by the fact that one must take into account the reward on average which investors collect from the underlying asset (see [5]).

Since the stochastic equation in (4.3) admits an explicit solution given by

$$X_t = X_0 e^{\varsigma \mathbb{B}_t + (r - \frac{1}{2} \varsigma^2) t}, \quad t \geq 0,$$

and since this solution is simple enough, it is possible to implement the recursive procedure from (3.1) with a single iteration. This means that, in this special case, the Feynman–Kac formula in (3.2) is directly computable without the need to approximate the integral kernel of the associated Markovian semigroup. This

also eliminates the need for interpolation, if all that one wants to do is compute the solution $u[t, x]$ for some fixed t and some fixed x . Of course, this assumes that the payoff can be expressed in a way that *Mathematica* can understand. In our first example, we will compute $u[0, 50]$, assuming that $\varsigma = 0.2$, $r = 0.1$ and that the boundary condition is $u[3, x] = \Lambda[x]$, where $\Lambda[x] = \text{Max}[x - 40, 0]$.

`In[1]:= T = 3.0;  $\varsigma$  = 0.2; r = 0.1;  $\Lambda[x\_]$  := Max[x - 40, 0];`

`In[2]:= OptionPrice[S_] :=`

$$\frac{e^{-T \times r}}{\sqrt{2 \pi}} \text{NIntegrate}\left[\Lambda\left[S \times e^{\left(r - \frac{\varsigma^2}{2}\right) \times T + \varsigma \sqrt{T} y}\right] \times e^{-\frac{y^2}{2}}, \{y, -\infty, \infty\},\right.$$

`MaxRecursion -> 15, WorkingPrecision -> 20, PrecisionGoal -> 6]`

Here is the result.

`In[3]:= OptionPrice[50] // Timing`

`Out[3]= {0.13 Second, 20.7449}`

What we have just computed was the price of an European call option with a strike price of \$40 and $T = 3$ years to maturity, assuming that the current price of the underlying asset is \$50, that the volatility in the stock price is 20 percent/year and that the fixed interest rate is 10 percent/year. Of course, the price of such an option can be computed directly by using the Black–Scholes formula. First, we will express the Black–Scholes formula in symbolic form

$$\text{In[4]:= } \mathcal{N}[x_]:= \frac{1}{2} \text{Erf}\left[\frac{x}{\sqrt{2}}\right] + \frac{1}{2};$$

$$\begin{aligned} \text{BS}[r_ , \varsigma_ , x_ , K_ , T_] := & x \times \mathcal{N}\left[\left(\text{Log}\left[\frac{x}{K}\right] + \left(r + \frac{1}{2} \varsigma^2\right) \times T\right) / (\varsigma \sqrt{T})\right] - \\ & K \times e^{-r \times T} \mathcal{N}\left[\left(\text{Log}\left[\frac{x}{K}\right] + \left(r - \frac{1}{2} \varsigma^2\right) \times T\right) / (\varsigma \sqrt{T})\right] \end{aligned}$$

and then compute the price of the same option.

`In[5]:= BS[r,  $\varsigma$ , 50, 40, T] // Timing`

`Out[5]= {0. Second, 20.7449}`

One can see from this example that, from a practical point of view, the Feynman–Kac representation of the solution of the Black–Scholes PDE in *Mathematica* is just as good as the “explicit” Black–Scholes formula. (Ironically, the Black–Scholes formula takes somewhat longer to type, not to mention the time it might take to retrieve it from a book.) However, the Feynman–Kac representation is much more universal. Indeed, with no extra work—and no additional complications whatsoever—it allows one to compute the value of an option with a very general payoff. Consider for example an option that pays at maturity the smallest of the payoffs from a call with a strike price of \$40 and a put with a strike price of \$80.

`In[6]:=  $\Lambda[x\_]$  := Min[Max[x - 40, 0], Max[80 - x, 0]]`

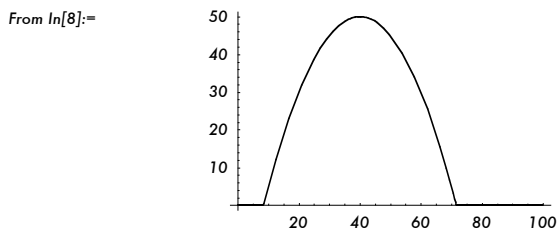
With all other parameters kept the same as before (3 years left to maturity, 20 percent volatility and 10 percent interest rate), assuming that the stock price is \$50, the price of this option is

```
In[7]:= OptionPrice[50] // Timing
```

```
Out[7]= {0.031 Second, 5.22782}
```

Obviously, the computing time is negligible, but what is much more important is that the *only* thing that we had to do in this case was to simply change the definition of the payoff. Here is another example.

```
In[8]:=  $\Lambda[x\_]:= \text{Max}[50 - 0.05 (x - 40)^2, 0]; \text{Plot}[\Lambda[x], \{x, 0, 100\}]$ 
```



```
In[9]:= OptionPrice[50] // Timing
```

```
Out[9]= {0.015 Second, 16.5291}
```

We will conclude this section with an example of what is known as a cash-or-nothing option.

```
In[10]:=  $\Lambda[x\_]:= 20 \times \text{Max}[\text{Sign}[x - 40], 0]$ 
```

```
In[11]:= Off[NIntegrate::"slwcon"]; OptionPrice[50] // Timing
```

```
Out[11]= {0.031 Second, 13.4738}
```

The overwhelming majority of the options that are currently traded are some combination of calls, puts, and cash-or-nothing options. The origins of this tradition are not exactly certain, but it might have something to do with the fact that these are the options for which popular and easy-to-use computing tools have been widely available. It should be clear from the examples presented in this section that, in the realm of the Black–Scholes model, essentially any reasonably defined option is, in fact, trivial to price, and that having a Black–Scholes formula is no longer necessary. Of course, the well-known binomial model does allow one to compute the value of an option with a very general payoff; however, this popular approach does require some work, and its accuracy and overall efficiency is far from satisfactory. Furthermore, this method is tied exclusively to the assumption that stock prices follow the law of a geometric Brownian motion; the technology that we are developing is not.

One must realize that, in keeping with the method outlined in Section 3, solving the PDE in (4.2) is an entirely trivial matter. The point is that the integral kernel of the associated Markovian semigroup happens to be directly computable, in the sense that it can be written in symbolic form that can be understood by *Mathemat-*

ica. However, our main goal is to demonstrate that this same method is just as easy to implement in much more general situations. In terms of applications to computational finance, that would mean that considerably more sophisticated models for asset values—for example, models with stochastic volatility and stochastic return of a rather general form—would be just as easy to use as the basic Black–Scholes model. The main step in this program is to develop an adequate approximation of the sample path representation of the semigroup $e^{t\mathcal{L}}$, $t \geq 0$, for a general elliptic differential operator $\mathcal{L}$ (see the next section).

■ 5. Approximation of the Fundamental Solution by Way of a Discrete Dynamic Programming Procedure

In general, there are many ways in which the solution of a given SDE can be approximated by a discrete time series (see [6] for a thorough exploration and a comprehensive list of references on this topic). Unfortunately, none of the available approximation schemes was developed for a purpose similar to ours, so we will develop one which actually is. Another extremely interesting approximation method, whose computational aspects are yet to be explored, can be found in [7].

To begin with, notice that the stochastic equation in (2.6) can be written in integral form as

$$X_t = x + \int_0^t \sigma[X_s] d\beta_s + \int_0^t a[X_s] ds, \quad X_0 = x \in \mathbb{R}, \quad t \geq 0. \quad (5.1)$$

One possible approximation of this equation can be obtained by replacing the coefficients $\sigma[\cdot]$ and $a[\cdot]$ by constants, which gives the following SDE

$$Y_t = x + \int_0^t \sigma[x] d\beta_s + \int_0^t a[x] ds, \quad Y_0 = x \in \mathbb{R}^n.$$

This approximation, which we will call the 0th-order approximation, is so crude that the SDE it leads to admits a trivial solution:

$$Y_t = x + \sigma[x] \beta_t + a[x] t, \quad t \geq 0.$$

Intuitively, it should be clear that, when t is sufficiently small, the distribution of the random variable (r.v.) Y_t is not very far from that of the r.v. X_t . If we were to implement the plan outlined in Section 3 with this level of approximation of the Markovian semigroup associated with the equation $(\partial_t + \mathcal{L})u[t, x] = 0$, the recursive procedure in (3.1) would come down to

$$\begin{aligned} & u\left[T - i \frac{T-t}{k}, x\right] \\ &= (2\pi)^{-\frac{1}{2}} \int_{\mathbb{R}} u\left[T - (i-1) \frac{T-t}{k}, x + \sigma[x] \frac{T-t}{k} \gamma + a[x] \frac{T-t}{k}\right] e^{-\frac{\gamma^2}{2}} d\gamma, \\ & \quad i = 1, \dots, k. \end{aligned}$$

Assuming that $u[T - (i-1) \frac{T-t}{k}, \cdot]$ is the result of polynomial interpolation, the above integral is certainly computable for every choice of x : in fact, it only has to

be computed for $x \in \mathcal{P}$, where $\mathcal{P}$ stands for the set of interpolation nodes. One can show that as $k \rightarrow \infty$ and as the number of interpolation nodes increases to ∞ , $u[T - t, \cdot]$ converges to the actual solution uniformly in the interpolation region. However, this convergence is known to be rather slow. The 0th-order approximation of SDEs is as old as the theory of SDEs itself and can be traced back to the original works of Itô—in fact, this approximation is crucial for his construction of the stochastic integral. One may say that the 0th-order approximation is “optimal” if all that one needs to do is show the existence of solutions to SDEs and derive various properties of the associated probability distributions, in which case the speed of the convergence is not an issue. In our case, however, it is and one can do considerably better by using the 1st-order Taylor approximation of the coefficients $\sigma[\cdot]$ and $a[\cdot]$. Our plan, then, is to replace the stochastic equation in (5.1) with the following SDE

$$Z_t = x + \int_0^t \left(\sigma[x] + \sigma'[x] (Z_s^x - x) \right) d\mathbb{B}_s + \int_0^t \left(a[x] + a'[x] (Z_s^x - x) \right) ds, \quad t \geq 0,$$

which admits this explicit solution

$$\begin{aligned} Z_t = & e^{\sigma'[x] \mathbb{B}_t + (a'[x] - \frac{1}{2} \sigma'[x]^2) t} \\ & \times \left((a[x] - a'[x] x) \int_0^t e^{-\sigma'[x] \mathbb{B}_s - (a'[x] - \frac{1}{2} \sigma'[x]^2) s} d\mathbb{B}_s \right. \\ & \left. + (\sigma[x] - \sigma'[x] x) \int_0^t e^{-\sigma'[x] \mathbb{B}_s - (a'[x] - \frac{1}{2} \sigma'[x]^2) s} d\mathbb{B}_s + x \right) \\ & - \sigma'[x] \times (\sigma[x] - \sigma'[x] x) t. \end{aligned} \quad (5.2)$$

In spite of the “explicit” form of this solution, the density of the r.v. Z_t is still difficult to compute. Therefore, we need to develop yet another level of approximation. In order to approximate the random variable on the right side of (5.2) with something computable, notice that if the integrand in the first integral is replaced by the constant 1 this will produce an error of order $o(t)$ (a *small o* of t). The approximation of the second integral is somewhat more involved. A straightforward application of the Itô formula yields

$$\begin{aligned} & \int_0^t e^{-\sigma'[x] \mathbb{B}_s - (a'[x] - \frac{1}{2} \sigma'[x]^2) s} d\mathbb{B}_s \\ & = \frac{1}{\sigma'[x]} \left(1 - e^{-\sigma'[x] \mathbb{B}_t - (a'[x] - \frac{1}{2} \sigma'[x]^2) t} \right) \\ & \quad + \left(\sigma'[x] - \frac{a'[x]}{\sigma'[x]} \right) \int_0^t e^{-\sigma'[x] \mathbb{B}_s - (a'[x] - \frac{1}{2} \sigma'[x]^2) s} ds, \end{aligned}$$

which shows that

$$\begin{aligned} & \int_0^t e^{-\sigma'[x] \mathbb{B}_s - (a'[x] - \frac{1}{2} \sigma'[x]^2) s} d\mathbb{B}_s \\ & = \frac{1}{\sigma'[x]} \left(1 - e^{-\sigma'[x] \mathbb{B}_t - (a'[x] - \frac{1}{2} \sigma'[x]^2) t} \right) + \sigma'[x] t - \frac{a'[x]}{\sigma'[x]} t + o(t). \end{aligned}$$

Thus, (5.2) can be rewritten as

$$Z_t = e^{\sigma'[x] \mathbb{B}_t + (a'[x] - \frac{1}{2} \sigma'[x]^2) t} \times \left((a[x] - a'[x] x) t + (\sigma[x] - \sigma'[x] x) \right. \\ \left. \times \left(\frac{1}{\sigma'[x]} \left(1 - e^{-\sigma'[x] \mathbb{B}_t - (a'[x] - \frac{1}{2} \sigma'[x]^2) t} \right) - \frac{a'[x]}{\sigma'[x]} t \right) + x \right) + o(t).$$

Of course, this identity holds only if $\sigma'[x] \neq 0$, but this is the only truly interesting case, since $\sigma'[x] = 0$ implies that

$$Z_t = e^{a'[x] t} \times \left((a[x] - a'[x] x) t + \sigma[x] \int_0^t e^{-a'[x] s} d\mathbb{B}_s + x \right),$$

where the second integral is a Gaussian r.v. of variance $\frac{1}{2 a'[x]} (1 - e^{-2 a'[x] t}) = t - a'[x] t^2 + O(t^3)$. Thus, we will approximate expressions of the form $E[f(Z_t)]$ with

$$\frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f[\psi[t, x, \gamma]] e^{-\frac{\gamma^2}{2}} d\gamma \quad (5.3)$$

where

$$\ln[12] := \varepsilon = 0.0001; \psi[t_ , x_ , \gamma_] := \text{If}[\text{Abs}[\sigma'[x]] > \varepsilon,$$

$$e^{\sigma'[x] \times \gamma \times \sqrt{t} + (a'[x] - \frac{1}{2} \sigma'[x]^2) \times t} \times \left((a[x] - a'[x] \times x) \times t + (\sigma[x] - \sigma'[x] \times x) \times \right. \\ \left. \left(\frac{1}{\sigma'[x]} \left(1 - e^{-\sigma'[x] \times \gamma \times \sqrt{t} - (a'[x] - \frac{1}{2} \sigma'[x]^2) \times t} \right) - \frac{a'[x]}{\sigma'[x]} \times t \right) + x \right), \\ \left. e^{a'[x] \times t} \times \left((a[x] - a'[x] \times x) \times t + \sigma[x] \times \gamma \times \sqrt{t - a'[x] \times t^2} + x \right) \right]$$

In particular, the recursive procedure in (3.1) will be replaced with

$$u\left[T - i \frac{T-t}{k}, x\right] \\ = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} u\left[T - (i-1) \frac{T-t}{k}, \psi\left[\frac{T-t}{k}, x, \gamma\right]\right] e^{-\frac{\gamma^2}{2}} d\gamma, \\ i = 1, \dots, k,$$

in the case of a terminal value problem, and with

$$u\left[T - i \frac{T-t}{k}, x\right] \\ = \text{Max}\left[\Lambda[x], \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} u\left[T - (i-1) \frac{T-t}{k}, \psi\left[\frac{T-t}{k}, x, \gamma\right]\right] e^{-\frac{\gamma^2}{2}} d\gamma\right], \\ i = 1, \dots, k,$$

in the case of a free boundary value problem. In Section 6 we will show that the computation in *Mathematica* of the last two integrals for a fixed $x \in \mathcal{P}$ is com-

pletely straightforward when the object $u[T - (i - 1) \frac{T-t}{k}, \cdot]$ is the result of polynomial interpolation.

It is not hard to check that in the special case where $\sigma[x] \equiv \varsigma \times x$ and $a[x] \equiv r \times x$ one has

$$\psi[\Delta, x, y] = x e^{\varsigma y \sqrt{\Delta} + (r - \frac{1}{2} \varsigma^2) \Delta},$$

which means that, with this choice for $\sigma[\cdot]$ and $a[\cdot]$, the solution to the stochastic equation in (5.1) is simply $X_t = \psi[t, x, \mathfrak{B}_t]$.

■ 6. Dynamic Programming with *Mathematica*: The Case of American and European Stock Options

First, we will consider several examples of terminal-value problems of the form:

$$(\partial_t + \mathcal{L}) u[t, x] = 0, \quad x \in \mathbb{R}, \quad t \leq T, \quad u[T, x] = \Lambda[x] \quad (6.1)$$

where $\mathcal{L}$ stands for the differential operator $\frac{1}{2} \sigma^2[x] \partial_{xx} + a[x] \partial_x - r$. The method that we developed in Sections 3 and 5 can be implemented in *Mathematica* with the following three definitions (in addition to the definition of $\psi[t, x, \gamma]$ from Section 5), in which the interpolation region is the interval $[\Sigma - b, \Sigma + b]$, the number of (equally spaced) interpolation nodes is $2m + 1$, and the time step is $\frac{T-t}{k}$. The object $U[i, \cdot, k, \cdot, \cdot, \cdot]$ represents the list of all $2m + 1$ values of $u[T - i \frac{T-t}{k}, \cdot]$ at the entire set of interpolation nodes $\mathcal{P}$ and, for a given object f , which *Mathematica* treats as a function, $A[f, \cdot, \cdot, \cdot, \cdot]$ returns the list of values of $E[f[\psi[d, x, \gamma]]]$ for all $x \in \mathcal{P}$.

$$\text{In}[13]:= U[0, T_ , k_ , \Sigma_ , h_ , m_] := \text{Table}[\Lambda[\Sigma + j \times \frac{b}{m}], \{j, -m, m\}]$$

$$\text{In}[14]:= U[i_ , T_ , k_ , \Sigma_ , h_ , m_] :=$$

$$\left(f[x_] = \text{ListInterpolation}[U[i - 1, T, k, \Sigma, b, m], \{\{\Sigma - b, \Sigma + b\}\}][x]; \right. \\ \left. \text{Table}[\frac{e^{-rx \frac{T}{k}}}{\sqrt{2\pi}} \times \text{NIntegrate}[f[\psi[\frac{T}{k}, \Sigma + j \times \frac{b}{m}, y]] e^{\frac{-y^2}{2}}, \right. \\ \left. \{y, -7, 7\}, \text{MaxRecursion} \rightarrow 20], \{j, -m, m\} \right]$$

Note that we have replaced $\int_{-\infty}^{\infty}$ with $\int_{-7}^7$. This makes the integration procedure more efficient, while the loss in precision is well within the accuracy of the method. One must be aware, however, that the interval $[-7, 7]$ may be too narrow for some problems.

Since our goal here is not to improve *Mathematica*'s performance, we will ignore the following error messages.

```
In[15]:= Off[InterpolatingFunction::"dmval"]; Off[InterpolatingFunction::"dmvali"];
Off[NIntegrate::"ploss"]; Off[NIntegrate::"ncvb"];
Off[NIntegrate::"slwcon"]; Off[InverseFunction::"ifun"];
Off[Power::"infy"]; Off[$MaxExtraPrecision::"meprec"]; Off[∞::"indet"]
```

In our first example, we take

```
In[16]:= Λ[x_] := Max[x - 80, 0]; r = 0.1; σ[x_] := 0.2 × x; a[x_] := r × x;
```

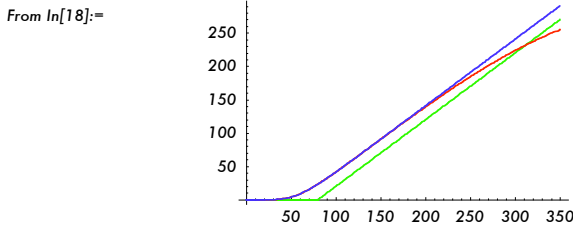
The solution $u[0, x]$ gives the value of an European call option with a strike price of \$80 that expires in T years, assuming that the volatility of the stock is 20 percent/year and the interest rate is 10 percent/year. We will take $T = 3$ years to maturity, $k = 5$ time periods (5 iterations), and 61 interpolation nodes ($m = 30$) in the interpolation interval $[[80 - 75, 80 + 75]]$.

```
In[17]:= μ = U[5, 3, 5, 80, 75, 30] // Timing; μ[1]
```

```
Out[17]= 2.172 Second
```

We can compare the result (the red curve) with the “exact” solution (the blue curve) produced by the Black–Scholes formula; the green curve represents the termination payoff.

```
In[18]:= Plot[{Λ[x], ListInterpolation[μ[2], {{80 - 75, 80 + 75}}][x],
BS[0.1, 0.2, x, 80, 3]}, {x, 0, 350}, PlotPoints → 200,
PlotRange → All, PlotStyle → {Hue[0.3], Hue[0.], Hue[.7]}]
```



One can see that the numerical solution (the red curve) is reasonably close to the exact solution in most of the interpolation region. To develop some sense for the accuracy of the method, we will compare the value of the option when the stock price is \$50, produced by the dynamic programming procedure

```
In[19]:= ListInterpolation[μ[2], {{80 - 75, 80 + 75}}][50]
```

```
Out[19]= 3.7509
```

with the following value produced by the Black–Scholes formula.

```
In[20]:= BS[0.1, 0.2, 50, 80, 3]
```

```
Out[20]= 3.75529
```

Of course, one can also compute this value by using the Feynman–Kac formula, just as we did in Section 4.

```
In[21]:= r = 0.1; σ = 0.2; OptionPrice[50]
```

```
Out[21]= 3.75529
```

As was pointed out earlier, when the coefficients are linear, $\psi[]$ gives the exact solution and one can get away with a single iteration, that is, with a time step equal to the entire time interval. Essentially this comes down to using the Feynman–Kac formula directly.

```
In[22]:=  $\mu = U[1, 3, 1, 80, 75, 30]; \text{ListInterpolation}[\mu, \{\{80 - 75, 80 + 75\}\}][50]$ 
```

```
Out[22]= 3.75083
```

The only reason the last two results are different is that $U[]$ applies the Feynman–Kac formula to the interpolated payoff, while $\text{OptionPrice}[]$ applies the Feynman–Kac formula to the actual payoff. Of course, there is no reason to use the dynamic programming procedure when the coefficients are linear and one deals with a fixed boundary value problem; here we did so only for the purpose of illustration.

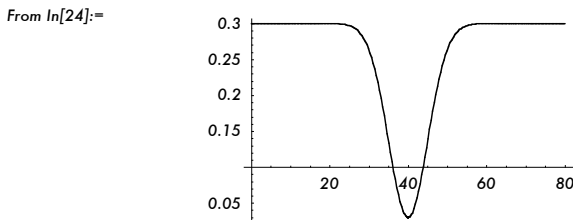
Now we will consider an example in which the coefficients $\sigma[]$ and $a[]$ are not linear, and therefore the Feynman–Kac formula cannot be used directly, that is, with a single iteration.

```
In[23]:=  $\sigma[x_] := 0.3 (1 - 0.9 e^{-0.02 (40-x)^2}) \times x; r = 0.1;$   

 $a[x_] := r \times x; \Lambda[x_] := \text{Max}[x - 40, 0]$ 
```

With this choice of the coefficients $\sigma[]$ and $a[]$ and the terminal data $\Lambda[]$, the solution of the PDE in (6.1) will give the price of a call option with a strike price of \$40, but with stochastic volatility for the underlying asset. For the purpose of illustration, we choose a volatility coefficient that depends on the price of the stock: it decreases to 0.1 when the price of the stock approaches the strike price of the option and stays constant at 0.2 when the stock price is away from the strike price.

```
In[24]:=  $\text{Plot}[\frac{\sigma[x]}{x}, \{x, 0.05, 80\}, \text{PlotRange} \rightarrow \text{All}, \text{PlotPoints} \rightarrow 200]$ 
```



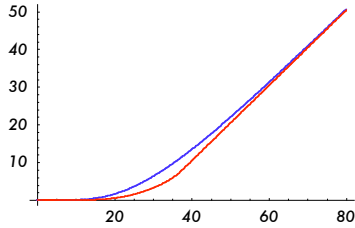
```
In[25]:=  $\text{mk} = U[20, 3, 20, 50, 48, 50] // \text{Timing}; \text{mk}[[1]]$ 
```

```
Out[25]= 14.828 Second
```

Now we can compare the price of this option with the one produced by the Black–Scholes formula (the blue line), which ignores the fact that the volatility decreases when the stock price approaches the strike price. Just as one would expect, when the stock price is away from the region where the volatility decreases, the price of the option looks the same as the price produced under the assumption of constant volatility.

```
In[26]:= Plot[{BS[0.1, 0.3, x, 40, 3], ListInterpolation[mk[[2]], {{50 - 48, 50 + 48}}][x]],
           {x, 0, 80}, PlotPoints -> 400, PlotStyle -> {Hue[0.7], Hue[0.]}
```

From In[26]:=



When the stock price is \$50, the price of this option is

```
In[27]:= ListInterpolation[mk[[2]], {{50 - 48, 50 + 48}}][50]
```

Out[27]= 20.4144

Notice that we have chosen the interpolation interval $[50 - 48, 50 + 48]$ so that it is centered around the stock price of interest, namely \$50; remember that the accuracy of the method is the highest at the center of the interpolation interval. Notice also that, while it took much longer to compute, the calculation of the option price under stochastic volatility did not require any additional work.

Our next example, which is unrelated to finance, will compute the solution $x \rightarrow u[0, x]$ of the equation

$$\partial_t u[t, x] + e^{-x^2} \partial_{x,x} u[t, x] + \text{Sin}[x] \partial_x u[t, x] = 0$$

with boundary condition $u[3, x] = \text{Sign}[x]$.

```
In[28]:= σ[x_] := Sqrt[2] e^(-x^2/2); a[x_] := Sin[x]; Λ[x_] := Sign[x]; r = 0;
```

The interpolation interval is $[0 - 5, 0 + 5]$, the number of interpolation nodes is $2 \times 25 + 1 = 51$, and the time step is $\frac{3}{30} = 0.1$.

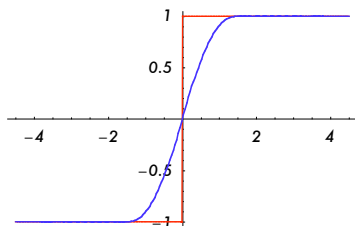
```
In[29]:= g0 = U[30, 3., 30, 0, 5, 25] // Timing; g0[[1]]
```

Out[29]= 10.656 Second

Now we can compare the solution that we just found, that is, the function $u[0, \cdot]$ (pictured as the blue line below) with the terminal data $u[3, \cdot]$ (pictured as the red line).

```
In[30]:= Plot[{Λ[x], ListInterpolation[g0[[2]], {{-5, 5}}][x]],
           {x, -4.5, 4.5}, PlotStyle -> {Hue[0], Hue[.7]}, PlotPoints -> 200]
```

From In[30]:=



Sometimes one must use the dynamic programming procedure in a nontrivial way and work with some small time step, even if the coefficients $\sigma[]$ and $a[]$ happen to be linear; such a case is an optimal stopping problem, which leads to boundary conditions prescribed along a free boundary. Now we will consider several examples that involve stock options in which early exercise is possible (the so-called American options). We only need to change the definition of the function $U[]$; everything else in the procedure will remain the same.

```
In[31]:= U[i_, T_, k_, Σ_, h_, m_] :=
  (f[x_] = ListInterpolation[U[i - 1, T, k, Σ, b, m], {{Σ - b, Σ + b}}][x]; Table[
    Max[Λ[Σ + j ×  $\frac{b}{m}$ ],  $\frac{e^{-r \times \frac{T}{k}}}{\sqrt{2\pi}} \times \text{NIntegrate}[f[\psi[\frac{T}{k}, \Sigma + j \times \frac{b}{m}, y]] e^{-\frac{y^2}{2}},$ 
      {y, -7, 7}, MaxRecursion → 20]], {j, -m, m}])
```

Under the Black–Scholes assumption for the price process, American call options are the same as European call options because early exercise of such options is never optimal. Thus, in our first example we consider an American put option with a strike price of \$40.

```
In[32]:= Λ[x_] := Max[40 - x, 0]
```

First, we will compute the price of the option under the assumption that—just as in the Black–Scholes model—the stock price follows a geometric Brownian motion process (of course, the Black–Scholes formula cannot be used in this case as is, for it is only valid for European options).

```
In[33]:= r = 0.1; σ[x_] := 0.2 × x; a[x_] := r × x;
```

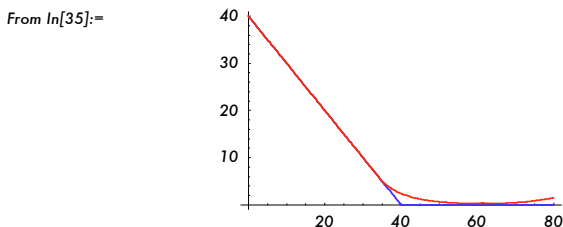
We will interpolate the solution in the interval $[40 - 38, 40 + 38] \equiv [2, 78]$ with 41 interpolation nodes and use 20 iterations, which amounts to a time step of 0.15.

```
In[34]:= m = U[20, 3, 20, 40, 38, 20] // Timing; m[[1]]
```

```
Out[34]= 6.438 Second
```

Now we can plot the solution (the red curve below) together with the termination payoff function (the blue curve).

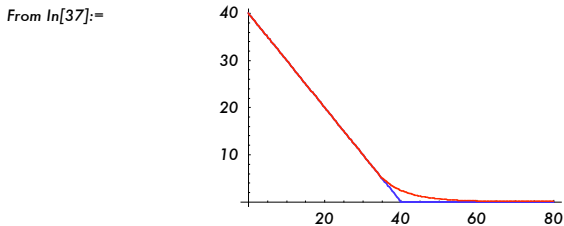
```
In[35]:= Plot[{Λ[x], ListInterpolation[m[[2]], {{40 - 38, 40 + 38}}][x]},
  {x, 0, 80}, PlotPoints → 200, PlotStyle → {Hue[0.7], Hue[0.]}
```



To those familiar with the basics of American options, it should be clear from this graph that the approximation that we just produced is not very accurate in the last 1/4 of the interpolation interval (it is well known that the actual stock price decreases to 0 monotonically). This is caused by the fact that, when the process X_t , $t \geq 0$, starts from a point that is close to the end of the interpolation interval, a sizeable part of the distribution of X_t might be spread in the region where the procedure is forced to use extrapolation. As a result, depending on how far inside the extrapolation region the essential part of the distribution of X_t goes, the error from the extrapolation can be significant. In this particular example, one possible remedy would be to set $u[T - (i - 1) \frac{T-t}{k}, x]$ to 0 for all sufficiently large values of x , regardless of the value produced by the interpolation (or the extrapolation) procedure. This is a common problem in essentially all numerical methods for PDEs, and the common way around it is to work with a grid in the state-space which covers a considerably larger region than the one in which an accurate representation of the solution is being sought. It is to be noted, however, that, as $k \rightarrow \infty$, that is, as the time step converges to 0, the solution produced after k iterations will converge to the actual solution in the *entire* interpolation interval. This is because when the process X_t , $t \geq 0$, starts from one of the end-points of that interval, for a very small value of t , the distribution of X_t will be concentrated mostly around the end-point where the extrapolation procedure is still reasonably accurate. Thus, by using the method developed here, one can construct an arbitrarily accurate representation of the solution in *the entire interpolation interval* by simply choosing a sufficiently large number of interpolation nodes and a sufficiently small time step. In general, this is not true for the finite difference method. In order to illustrate this phenomenon, we will now repeat the above calculation with a time step, which is three times as small (accordingly, the calculation will run about three times longer).

```
In[36]:= mm = U[60, 3, 60, 40, 38, 20] // Timing; mm[[1]]
Out[36]:= 14.89 Second

In[37]:= Plot[{Λ[x], ListInterpolation[mm[[2]], {{40 - 38, 40 + 38}}][x]},
             {x, 0, 80}, PlotPoints -> 200, PlotStyle -> {Hue[0.7], Hue[0.]}]
```



Let us remember that here we are also solving an optimal stopping problem, whose solution is given by the point, which separates the “exercise” region, that is, the region where the price of the option coincides with the termination payoff, from the “hold” region, where the price of the option is larger than the termination payoff.

```
In[38]:= FindRoot[ListInterpolation[mm[[2]], {{40 - 38, 40 + 38}}][x] == Max[40 - x, 0],
             {x, 38, 40}]
```

```
Out[38]= {x -> 34.3}
```

It is interesting that the solution, which was produced earlier and about three times faster by using a three times larger time step, gives the same result.

```
In[39]:= FindRoot[ListInterpolation[m[[2]], {{40 - 38, 40 + 38}}][x] == Max[40 - x, 0],
             {x, 38, 40}]
```

```
Out[39]= {x -> 34.3}
```

This illustrates perfectly the point that we made earlier: the result produced with the larger time step was already reasonably accurate in the middle of the interpolation interval.

Now we will compute the price of the same put option, but under the assumption that the stock price exhibits stochastic volatility of the type we introduced earlier in conjunction with European-type options.

```
In[40]:=  $\sigma[x_] := 0.2 \times (1 - 0.5 e^{-0.01 \times (40 - x)^2}) \times x;$ 
           $r = 0.1;$   $a[x_] := r \times x;$   $\Lambda[x_] := \text{Max}[40 - x, 0]$ 
```

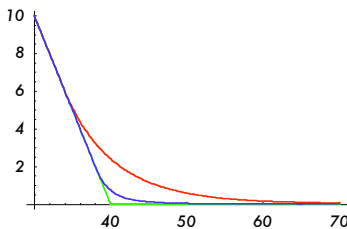
```
In[41]:= mx = U[60, 3, 60, 40, 38, 30] // Timing; mx[[1]]
```

```
Out[41]= 26.547 Second
```

Now we can compare the last result (the blue curve below) with the one that was produced in the case of constant volatility (the red curve).

```
In[42]:= Plot[{ $\Lambda[x]$ , ListInterpolation[mm[[2]], {{40 - 38, 40 + 38}}][x],
             ListInterpolation[mx[[2]], {{40 - 38, 40 + 38}}][x]}, {x, 30, 70},
             PlotPoints -> 200, PlotStyle -> {Hue[0.3], Hue[0.], Hue[0.7]}]
```

From In[42]:=



We can see that, just as one would expect, the price of this option differs from the one produced under the constant volatility assumption only in the region where the volatility decreases. Also, the decrease of the volatility near the exercise price actually decreases the range of stock prices for which immediate exercise is optimal, as the following computation shows.

```
In[43]:= FindRoot[ListInterpolation[mx[[2]], {{40 - 38, 40 + 38}}][x] == Max[40 - x, 0],
             {x, 23, 30}]
```

```
Out[43]= {x -> 30.}
```

■ 7. Concluding Remarks

Although the examples presented in Section 6 were borrowed mostly from the realm of computational finance, it is important to realize that neither the general method developed in Sections 3 and 5, nor the *Mathematica* code used in Section 6 takes advantage of the special nature of the problems associated with financial applications in any way. The same method—and, indeed, the same computer code—can produce an approximate solution to a rather general one-dimensional parabolic PDE with boundary conditions imposed on some free or fixed boundary. Furthermore, this method does not require any manipulation whatsoever of the PDE.

As is well known, the “no free lunch” principle reigns in the world of computing, too, and the simplicity, the generality, and the precision with which we were able to solve some classical option valuation problems in the previous section did not come easily by any means. In terms of this metaphor, not only was the lunch not free, but it was, in fact, a rather expensive one. Indeed, an efficient integration procedure was absolutely crucial for the implementation of the method in Section 6. As explained in *The Mathematica Book* (see [8] p. 1071) the function `Integrate[]` “uses about 500 pages of *Mathematica* code and 600 pages of C code.” The role played by the function `ListInterpolation[]` was just as crucial. Indeed, it allowed us to work on an exceptionally coarse grid in the state-space and then “restore” the values at all remaining points by a fairly sophisticated interpolation procedure. More importantly, we were able to feed `NIntegrate[]` with the object produced by `ListInterpolation[]`. There are intrinsic obstacles to such “shortcuts” in essentially all modifications of the finite difference method.

■ Acknowledgment

This work is dedicated to D. W. Stroock and S. R. S. Varadhan. All numerical procedures used here are a mere transcript of their insights.

■ References

- [1] F. Black and M. Scholes, “The Pricing of Options and Corporate Liabilities,” *Journal of Political Economy*, **81**, 1973 pp. 637–659.
- [2] D. W. Stroock and S. R. S. Varadhan, “Multidimensional Diffusion Processes,” *Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*, **233**, New York, Berlin: Springer-Verlag, 1979.
- [3] D. W. Stroock, “Lectures on Stochastic Analysis: Diffusion Theory,” *London Mathematical Society Student Texts*, **6**, 1987, Oxford: Cambridge University Press.
- [4] D. W. Stroock, *Probability Theory, an Analytic View*, Oxford: Cambridge University Press, 1993.
- [5] A. Lyasoff, “Three Page Primer in Financial Calculus.” (2000) Boston: Boston University. www.bu.edu/mathfn/people/primer1.pdf.

- [6] P. E. Kloeden and E. Platen, *Numerical Solutions of Stochastic Differential Equations*, New York: Springer-Verlag, 1997.
- [7] D. Stroock and S. Taniguchi, "Diffusions as Integral Curves, or Stratonovich without Itô," in *Progress in Probability #34*, ed. by M. Freidlin, Birkhäuser, 1994, pp. 331–369.
- [8] S. Wolfram, *The Mathematica Book*, 5th ed. Champaign, Oxford: Wolfram Media/Cambridge University Press, 1999.

About the Author

Andrew Lyasoff is associate professor and director of the M.A. degree program in Mathematical Finance at Boston University. His research interests are mostly in the area of Stochastic Calculus and Mathematical Finance. Over the last few years he has been using *Mathematica* extensively to develop new numerical methods for PDEs and new instructional tools for graduate courses in Operations Research and Mathematical Finance. In addition, he is developing new theoretical models that reflect long-range dependencies in market data.

Andrew Lyasoff

Boston University
Department of Mathematics and Statistics
Boston, MA
alyasoff@bu.edu

AuthorTools: A Package for Document Processing

Pavi Sandhu

AuthorTools is an add-on package that simplifies the task of creating publication-quality books and articles from notebooks. This article reviews the features of *AuthorTools* and provides detailed information on using its functions and palettes.

■ Introduction

Mathematica notebooks are structured documents that integrate text, graphics, code, and mathematical formulas. Notebooks are portable and platform-independent and can be easily transformed to other formats, such as HTML, $\text{\TeX}$, XML, or PDF. These features make notebooks a popular format for creating and publishing technical documents.

Although notebooks are typically created using *Mathematica*'s front end, they can also be created and modified using kernel commands. Each notebook is completely specified by a symbolic expression. You can, therefore, select and manipulate parts of a notebook by modifying the underlying expression. This makes it possible to write programs and packages for transforming notebook documents.

The *AuthorTools* package, which was introduced in Version 4.2, provides a good illustration of *Mathematica*'s document-processing abilities. This package adds support for several features that are important in technical publishing, such as creating a table of contents or index. It greatly simplifies the task of creating publication-quality books and articles from notebooks. For example, using *AuthorTools*, you can:

- create a table of contents
- create an index
- create a Browser Categories file for adding information to the Help Browser
- find all differences between any two notebooks
- create bilaterally formatted cells for displaying examples of *Mathematica* calculations

- extract all cells of a particular type and save them in a desired format
- insert objects to display the current values of variables, such as the date, time, and filename
- set printing options, such as headers and footers

Most of these operations can be done either on a single notebook or a set of notebooks that make up a project. For example, you can generate a unified index or table of contents for a book, each chapter of which is in a separate notebook.

The *AuthorTools* package contains about sixty functions and over ten palettes. The palettes provide an easy point-and-click interface for performing most common tasks. However, advanced users might prefer to type and evaluate the functions directly, since this allows for greater control and customization. The rest of this article reviews the various functions and palettes in more detail.

■ Functions

□ Introduction

Most *AuthorTools* functions require you to specify a notebook as the first argument. In general, you can specify a notebook either as a notebook object or a notebook file.

Notebook Objects

The *Mathematica* kernel refers to any open notebook via an expression of the form `NotebookObject[fe, id]`, where *fe* specifies the front end in which the notebook is open and *id* is a unique serial number for the notebook.

The command `Notebooks[]` returns a list of notebook objects corresponding to all notebooks currently open in the front end.

```
In[1]:= Notebooks[]

Out[1]= {NotebookObject[<<AuthorToolsGuide.nb>>],
        NotebookObject[<<Messages>>]}
```

This assigns the symbol `nb` to represent the first notebook object in the list.

```
In[2]:= nb = %[[1]]

Out[2]= NotebookObject[<<AuthorToolsGuide.nb>>]
```

Notebook Files

Alternatively, you can identify a notebook by specifying the name and location of the notebook file. You can use the `ToFileName` function to construct a string specifying the full pathname of the file.

```
In[3]:= nb = ToFileName[{$InstallationDirectory, "AddOns", "Applications",
                        "AuthorTools", "Documentation", "English"}, "AuthorToolsGuide.nb"]

Out[3]= C:\Program Files\Wolfram Research\Mathematica\5.0\AddOns\
        Applications\AuthorTools\Documentation\English\AuthorToolsGuide.nb
```

Loading the Package

This loads the *AuthorTools* package.

```
In[4]:= << AuthorTools`
```

You can then use the symbol `nb` as an argument to any notebook function.

```
In[5]:= NotebookName[nb]
```

```
Out[5]= AuthorToolsGuide.nb
```

Processing Multiple Notebooks

You can use *AuthorTools* functions to operate either on a single notebook or multiple notebooks in the directory.

To process a single notebook:

Specify the notebook as the first argument of the function you want to use.

In general, a function that acts on notebooks can accept either a notebook object or a notebook file as the first argument. The exceptions are those functions that act on the current selection in an open notebook, for example: `HorizontalInsertionPointQ`, `AddIndexEntry`, `SelectionRemoveCellTags`, or `IndexCellOnSelection`. These functions can accept a notebook object as the first argument but not a notebook file.

To process multiple notebooks:

1. Create a project file that lists all the notebooks you want to process. (See “Creating a Project File” for more information.)
2. Specify the project file as the first argument of the function you want to use. Alternatively, load the project file using the `MakeProject` dialog box, then use the notebook object corresponding to the dialog box as the argument of the function.

In general, a function that accepts a notebook file will also accept a project file. The exceptions are those functions that are only defined for a single notebook and hence cannot be applied to a project, such as: `NotebookName`, `NotebookFolder`, `NotebookCellTags`, or `NotebookFileOptions`. These functions can accept a notebook file as the first argument but not a project file.

□ General Notebook Manipulation

Getting Notebook Information

This gives the full pathname of the notebook object.

```
In[6]:= NotebookFilePath[nb]
```

```
Out[6]= C:\Program Files\Wolfram Research\Mathematica\5.0\AddOns\
        Applications\AuthorTools\Documentation\English\AuthorToolsGuide.nb
```

This gives the name of the notebook.

```
In[7]:= NotebookName[nb]
```

```
Out[7]= AuthorToolsGuide.nb
```

This gives the directory in which the notebook is located.

```
In[8]:= NotebookFolder[nb]
```

```
Out[8]= C:\Program Files\Wolfram Research\Mathematica\5.0\
        AddOns\Applications\AuthorTools\Documentation\English\
```

This gives all front end options set in the notebook.

```
In[9]:= NotebookFileOptions[nb]
```

```
Out[9]= {FrontEndVersion → 5.0 for Windows,
        ScreenRectangle → {{0, 1112}, {0, 746}},
        ScreenStyleEnvironment → Brackets,
        PrintingStyleEnvironment → EnhancedPrintout,
        WindowToolbars → {}, WindowSize → {644, 719},
        WindowMargins → {{Automatic, 14}, {Automatic, 1}},
        PrintingCopies → 1, PrintingPageRange → {1, Automatic},
        PrintingOptions → {GraphicsPrintingFormat → RenderInFrontEnd},
        DefaultNewCellStyle → Text, Magnification → 1,
        StyleDefinitions → HelpBrowser.nb}
```

<code>NotebookFilePath[nb]</code>	the full path to the specified notebook
<code>NotebookName[nb]</code>	the filename of the specified notebook
<code>NotebookFolder[nb]</code>	the directory of the specified notebook

Some notebook functions.

Using the Notebook Cache

Any notebook file created in the front end has a cache that contains information about the notebook's cell structure. The cache appears at the end of the notebook file and can be inspected by opening the notebook in a text editor.

The notebook's cache contains information on the position, style, grouping, and size of each cell. When you open an existing notebook, the front end uses the information in the cache to read into memory only those cells that need to be displayed on the screen. This reduces the memory and time required to read and render a notebook since the entire notebook does not have to be read into memory at once.

The main component of the cache is a notebook file outline. This has the same form as the notebook expression describing the notebook but with each cell expression replaced by a condensed version. Instead of the actual contents of the cell, the condensed expression contains information on the cell's position in the notebook and the number of bytes of data it contains.

Each condensed expression is of the form: `Cell[a, b, c, d, e, rules, string]`. The parameters *a* through *e* are integers and denote the following:

a = stream position of the `Cell[]` expression, counting all newlines as 1 byte long

b = number of newlines in the file before the `Cell[]` expression

c = length of `Cell[]` in bytes, counting all newlines as 1 byte long

d = number of newlines in `Cell[]`

e = approximate pixel height in cell (used in determining the full height of the notebook for purposes of scaling the scrollbar thumb)

You can view the file outline in a notebook's cache using the `NotebookFileOutline` line function.

```
In[10]:= NotebookFileOutline[nb]
```

```
Out[10]= Notebook[{Cell[CellGroupData[{Cell[1727, 52, 22, 0, 108, Title],
    Cell[CellGroupData[{Cell[1774, 56, 28, 0, 56, Section],
        Cell[CellGroupData[{Cell[1827, 60, 34, 0, 46, Subsection],
            Cell[1864, 62, 27, 0, 30, Text]}, Open]],
        Cell[1906, 65, 34, 0, 46, Subsection]}, Open]],
    Cell[1955, 68, 28, 0, 56, Section]}, Open]]}]
```

The `NotebookLookup` function extracts information from the notebook file outline. With "CellOutline" as the second argument of `NotebookLookup`, you get a list of all the condensed cell expressions occurring in the notebook file outline.

```
In[11]:= NotebookLookup[nb, "CellOutline"]
```

```
Out[11]= {Cell[1727, 52, 22, 0, 108, Title], Cell[1774, 56, 28, 0, 56, Section],
    Cell[1805, 58, 36, 0, 46, Subsection],
    Cell[1844, 60, 36, 0, 46, Subsection],
    Cell[1895, 63, 28, 0, 56, Section], Cell[1926, 65, 28, 0, 56, Section]}
```

With "CellExpression" as the second argument of `NotebookLookup`, you get the complete cell expression for each cell in the notebook.

```
In[12]:= NotebookLookup[nb, "CellExpression"]
```

```
Out[12]= {Cell[Title, Title], Cell[Section 1, Section],
    Cell[Subsection 1.1, Subsection], Cell[Subsection 1.2, Subsection],
    Cell[Section 2, Section], Cell[Section 3, Section]}
```

With "CellIndex" as the second argument of `NotebookLookup`, you get the serial number of each cell in the notebook. The first cell in the notebook has serial number 1 and so on.

```
In[13]:= NotebookLookup[nb, "CellIndex"]
```

```
Out[13]= {1, 2, 3, 4, 5, 6}
```

The two-argument form of `NotebookLookup` simply generates a list of the form $\{1, 2, \dots, n\}$ for a notebook containing *n* cells. The `NotebookLookup` command

becomes more useful if you use the optional third argument to specify a pattern. This allows you to extract information on all cells that fit the pattern. For example, the following command gives the serial number of all Section cells.

```
In[14]:= NotebookLookup[nb, "CellIndex", Cell[___, "Section", ___]]
```

```
Out[14]= {1, 14, 245, 385}
```

The following command returns the cell expressions of all Section cells.

```
In[15]:= NotebookLookup[nb, "CellExpression", Cell[___, "Section", ___]]
```

```
Out[15]= {Cell[What is AuthorTools?, Section, CellTags → What is AuthorTools?],
          Cell[Packages, Section],
          Cell[Functions List, Section], Cell[Palettes, Section]}
```

<code>NotebookFileOutline[nb]</code>	returns the notebook file outline cache for the specified notebook
<code>NotebookLookup[nb, obj]</code>	returns the objects specified by <i>obj</i> in the specified notebook
<code>NotebookLookup[nb, obj, pat]</code>	returns the objects specified by <i>obj</i> that match the pattern <i>pat</i> in the specified notebook
<code>NotebookFileOptions[nb]</code>	the front end options set in the specified notebook

Some functions that provide information about a notebook's contents.

Adding and Removing Cell Tags

A cell tag is a string that serves as a tag or marker to identify a specific cell in a notebook. Cell tags are used, for example, to define the targets of hyperlinks or to identify the information that appears when you choose a topic in the Help Browser. *AuthorTools* includes functions for adding or removing cell tags in a notebook.

`NotebookCellTags` returns a list of all cell tags in the specified notebook. The *n*th element of the output list is a list of all cell tags for the *n*th cell in the notebook. Each cell that does not have a cell tag is represented by an empty list.

```
In[16]:= NotebookCellTags[nb]
```

```
Out[16]= {{}, {}, {}, {}, {}, {}}
```

You can add cell tags to cells in a notebook using `AddCellTags`.

You can remove cell tags from cells in a notebook using `RemoveCellTags`.

<code>AddCellTags[nb]</code>	adds the string or list of strings to the cell tags of the currently selected cell(s) in the notebook object <i>nb</i>
<code>RemoveCellTags[nb, pat]</code>	removes all cell tags from the notebook object <i>nb</i> that match the string pattern <i>pat</i>
<code>SelectionRemoveCellTags[nb, tags]</code>	removes the specified tag or list of tags from the currently selected cell(s) in the notebook object <i>nb</i>
<code>NotebookCellTags[nb]</code>	returns a list of all the cell tags present in the notebook object <i>nb</i>

Some functions for adding and removing cell tags.

□ Processing Multiple Notebooks

Creating a Project File

The first step in processing multiple notebooks is to create a project file. This is a file that specifies the names and location of all notebooks in a project. You will need to create a different project file for each project you are working on.

A project file is a plain text file with a `.m` suffix. The file must include data in the following format.

```
{ "Name" -> name,
  "Directory" -> directory,
  "Files" -> {nb1, nb2, ...} }
```

Here, *name* is the name of the project, *directory* is the full pathname of the project directory, and {*nb1*, *nb2*, ...} is a list of all notebooks that make up the project.

There are several different ways to create a project file:

- Enter the project data in the text fields of the MakeProject dialog box. (See MakeProject for details.)
- Use the command `WriteProjectData[file, data]`, where *data* specifies the project data as specified in the preceding box.
- Type the project data directly into a text file and save the file with a `.m` suffix.

Using a Project File

Once you have created a project file, you can operate on all notebooks in the project in one step. To do this you simply specify the project file as the argument to the *AuthorTools* function you want to use.

For example, to generate a table of contents for a project, you can evaluate the command, `MakeContents[projectfile,"Book"]`. The command for generating a table of contents for a single notebook is `MakeContents[nb,"Book"]`.

Editing a Project File

When you evaluate an *AuthorTools* function with a project file as an argument, *Mathematica* uses `Get` to read in the project file. Any *Mathematica* commands present in the project file are evaluated at this time. Thus, the project file is a convenient place to store commands that you want to apply to a project.

For example, suppose you prepend the following command to the contents of your project file.

```
Needs["AuthorTools`MakeContents`"];
SetOptions[MakeContents,
  SelectedCellStyles -> {"Chapter", "Heading", "Subheading"}];
```

This sets the option `SelectedCellStyles` of the *AuthorTools* function `MakeContents`. If you then use `MakeContents` to create a table of contents for that project, the option setting specified in the project file will be automatically used, overriding the default behavior of `MakeContents`.

Note: Any commands you have added to a project file will be overwritten if you use the `WriteProjectData` function or the `MakeProject` dialog box to modify the project file.

□ Creating a Table of Contents

This generates a table of contents for the notebook. The table of contents is saved in the same directory as the source notebook.

```
MakeContents[nb,"Simple"]
```

The second argument of the function specifies the format of the table of contents. You can choose from three different formats: `"Simple"`, `"Book"`, or `"BookCondensed"`. (See `MakeContents` for more information on these formats.)

Note: The `MakeContents` command inserts cell tags into the source notebook and automatically saves the changes. If you do not want your source notebook modified, you should keep a separate copy as a back-up.

If you are going to generate a table of contents in any format other than `Simple`, you should first use `Paginate`. This function calculates the page numbers for the specified notebook and stores them as `TaggingRules` in the notebook.

```
Paginate[nb]
```

The following command generates a table of contents in the `Book` format.

```
MakeContents[nb, "Book"]
```

The option `SelectedCellStyles` determines which cells in the notebook are included in the table of contents. The default setting is `SelectedCellStyles → {"Title", "Section", "Subsection", "Subsubsection"}`. The following command generates a table of contents in the Book format that includes only Title, Section, and Subsection cells.

```
MakeContents[nb, "Book",  
  SelectedCellStyles → {"Title", "Section", "Subsection"}]
```

□ Creating an Index

There are two steps involved in setting up an index.

1. Associate index entries with specific cells in your source notebook.

The simplest way to insert index entries is using the Edit Notebook Index dialog box, accessed from the MakeIndex palette. Alternatively, you can type and evaluate the `AddIndexEntry` function. The following command associates the specified main entry and subentry with the currently selected cell(s) in the notebook.

```
AddIndexEntry[nb, {main, sub}]
```

2. Generate the index using the `MakeIndex` function.

This generates an index for the notebook.

```
MakeIndex[nb, "Simple"]
```

The second argument of `MakeIndex` specifies the format of the index. You can choose from four different formats: `"Simple"`, `"Book"`, `"Two-Column"`, or `"BrowserIndex"`. (See `MakeIndex` for more information on these formats.)

Note: The `MakeIndex` command inserts cell tags into the source notebook and automatically saves the changes. If you do not want your source notebook modified, you should keep a separate copy as a back-up.

□ Finding Differences

This generates a notebook that lists the differences between the two notebooks, *nb1* and *nb2*.

```
NotebookDiff[nb1,nb2]
```

`NotebookDiff` will also find the differences between two projects, directories, or lists of files. This generates a notebook that summarizes the differences in the two sets of notebooks.

```
NotebookDiff[project1, project2]
```

By default, `NotebookDiff` finds all possible differences between the notebooks, including differences in cell styles or options. However, you can narrow the scope of differences reported by `NotebookDiff` by specifying options. For example, this excludes cells in the Input and Output style from the diffing operation.

```
NotebookDiff[nb1,nb2, ExcludeCellsOfStyles→ {"Input", "Output"}]
```

There are several other options to `NotebookDiff`, for example, to ignore cells that have the same content but differ only in their cell style or only in their options.

To view the differences within two different cells, use `CellDiff`. This highlights the differences in content by enclosing them in colored brackets of the form: (< and >). Style and option differences are also listed.

```
CellDiff[cell1, cell2]
```

If the notebooks you are comparing are style sheets, you can either use `NotebookDiff` or a more specialized version called `StyleSheetDiff`.

```
StyleSheetDiff[nb1,nb2]
```

`NotebookDiff` and `StyleSheetDiff` are both implemented in terms of a lower-level function called `DiffReport`. This compares two generic lists, creating a report of all the insertions, deletions, and updates between the two lists.

```
DiffReport[list1, list2]
```

□ Restoring a Notebook

NotebookRestore

`NotebookRestore` takes a notebook containing one or more syntax errors and creates a new notebook containing all the cells that did not have a syntax error. For example, suppose `nb1` represents a notebook file that has been corrupted by

removing one quote from a cell style name. If you try to open this file, the front end will report a syntax error and suggest you cancel the open operation.

```
In[17]:= nbObj = NotebookOpen[nbFile]
```

```
Out[17]= $Failed
```

Using `NotebookRestore`, you can at least access those cells in the notebook that are not corrupted. `NotebookRestore` will open a new notebook window containing all the good cells from the given notebook file and insert an indicator at each place where notebook data has been deleted.

```
In[18]:= NotebookRestore[nbFile]
```

```
Out[18]= NotebookObject[<<Untitled-1>>]
```

Instead of deleting the corrupt data, you can display it verbatim within the indicator by setting the option `DeleteCorruptCells` $\rightarrow$ `False`.

```
In[19]:= nbObj = NotebookRestore[nbFile, DeleteCorruptCells  $\rightarrow$  False]
```

```
Out[19]= NotebookObject[<<Untitled-2>>]
```

All the corrupt cell indicators have the cell tag "Corrupt", so you can easily highlight them by choosing `Corrupt` from the list of cell tags displayed under the `Find` $\triangleright$ `Cell Tags` menu. Alternatively, you can walk through each indicator one by one using the `NextCorruptCell` function.

```
In[20]:= NextCorruptCell[nbObj]
```

```
Out[20]= NotebookSelection[NotebookObject[<<Untitled-3>>]]
```

If your notebook contains large graphics cells or large blocks of typesetting, it may take a long time for *Mathematica* to read in the expression and try to determine if it contains any syntax errors. For that reason, there are options that allow you to skip over graphics and/or typeset cells without processing them.

```
In[21]:= Options[NotebookRestore]
```

```
Out[21]= {DeleteCorruptCells  $\rightarrow$  $DeleteCorruptCells,
          DeleteGraphicsCells  $\rightarrow$  $DeleteGraphicsCells,
          DeleteTypesetCells  $\rightarrow$  $DeleteTypesetCells}
```

```
In[22]:= NotebookRestore[nbFile, DeleteTypesetCells  $\rightarrow$  True]
```

```
Out[22]= NotebookObject[<<Untitled-4>>]
```

SalvageCells

`SalvageCells` is a noninteractive way to extract the list of cells from a corrupt notebook file. This function is intended for users who want to manipulate the cell listing directly, as part of a program. `SalvageCells` takes the same options, and returns the same information, as `NotebookRestore`.

This salvages all the cells from a notebook file.

```
In[23]:= cellList = SalvageCells[nbFile, DeleteCorruptCells  $\rightarrow$  False];
```

This gives the total number of cells.

```
In[24]:= Length[cellList]
```

```
Out[24]= 16
```

This gives the total number of blocks removed from the notebook.

```
In[25]:= Length[
    badCells = Cases[cellList, Cell[____, CellTags → "Corrupt", ____]]
```

```
Out[25]= 2
```

You can view just the bad data in a separate notebook.

```
In[26]:= NotebookPut[Notebook[badCells]]
```

```
Out[26]= NotebookObject[<<Untitled-5>>]
```

□ **Pagination**

This paginates the specified notebook. If the argument is a project file, the command paginates all notebooks in the project.

```
Paginate[nb]
```

This paginates all notebooks in a project. The setting `StartingPages → "Next"` means that the starting page for each notebook is the next page following the last page of the previous notebook.

```
Paginate[projectfile, StartingPages → "Next"]
```

Other possible settings for `StartingPages` are "Even", "Odd", `Inherited`, or a list of integers. The following command paginates all notebooks in a project and sets the starting page for the first four notebooks to be 1, 25, 57, and 83.

```
Paginate[projectfile, StartingPages → {1, 25, 57, 83}]
```

The following paginates all notebooks in a project. The starting page number for each notebook in the project is inherited from the setting of the option `StartingPageNumber` for that notebook.

```
Paginate[projectfile, StartingPages → Inherited]
```

□ **Creating a Browser Categories File**

This generates a simple browser categories file for the notebook.

```
MakeCategories[nb]
```

The second argument of `MakeCategories` specifies the format of the `Browser-Categories.m` file. You can choose from three different formats: "Simple", "Full", or "FullNoTags".

The following command generates a full browser categories file for the notebook.

```
MakeCategories[nb, "Full"]
```

Note: The `MakeCategories` command inserts cell tags into the source notebook and automatically saves the changes. If you do not want your source notebook modified, you should keep a separate copy as a back-up.

□ Creating Bilateral Cells

You can use the `MakeBilateral` function to create bilateral cells for displaying sample *Mathematica* calculations. Each bilateral cell consists of two columns with expository text to the left and input and output cells to the right. To create a bilateral cell, you must have a pair of input and output cells preceded by a cell containing text.

This is the integral of the sine function.

```
In[27]:= Integrate[Sin[x], x]
```

```
Out[27]= -Cos[x]
```

This converts the text cell and input-output pair into a bilateral cell.

```
In[28]:= MakeBilateral[nb]
```

Here is the resulting bilateral cell.

This is the integral of the
sine function.

```
In[1]:= Integrate[Sin[x], x]
Out[1]= -Cos[x]
```

The cell styles that appear in the left column of the bilateral cell are determined by `FirstBilateralStyles`, an option to `MakeBilateral`. The default setting of this option includes only one style, `MathCaption`.

The cell styles that appear in the right column of the bilateral cell are determined by `RestBilateralStyles`. The default setting includes `Input`, `Output`, and `Graphics` cell styles.

Once you have created a bilateral cell, you can reverse the process and get back the constituent cells using the following command.

```
In[29]:= DivideBilateral[nb]
```

□ Extracting Cells from a Notebook

This exports all Section style cells to a notebook.

```
In[30]:= ExportNotebook[nb, "Section", "Notebook"]
```

```
Out[30]= {/Applications/Mathematica 5.0.app/AddOns/
Applications/AuthorTools/Documentation/English/Sample11.nb}
```

You can also extract all Section cells by specifying a list of two elements as the second argument. `ExportNotebook[nb1, {"CellStyle", "Section"}, "Notebook"]` is equivalent to `ExportNotebook[nb1, "Section" "Notebook"]`.

```
In[31]:= ExportNotebook[nb, {"CellStyle", "Section"}, "Notebook"]
```

```
Out[31]= {/Applications/Mathematica 5.0.app/AddOns/
Applications/AuthorTools/Documentation/English/Sample11.nb}
```

This command exports all cell groups in which the heading cell is an input cell. This enables you, for example, to extract all input-output pairs in a notebook.

```
In[32]:= ExportNotebook[nb, {"CellGroup", "Input"}, "Notebook"]
```

```
Out[32]= {/Applications/Mathematica 5.0.app/AddOns/
Applications/AuthorTools/Documentation/English/Sample11.nb}
```

This command exports all cells having the cell tag “Reference1”.

```
In[33]:= ExportNotebook[nb, {"CellTag", "Reference1"}, "Notebook"]
```

```
Out[33]= {/Applications/Mathematica 5.0.app/AddOns/
Applications/AuthorTools/Documentation/English/Sample11.nb}
```

This command exports all graphics cells as separate GIF files.

```
In[34]:= ExportNotebook[nb, {"CellTag", "reference1"}, "GIF"]
```

```
Out[34]= {/Applications/Mathematica 5.0.app/AddOns/
Applications/AuthorTools/Documentation/English/Sample11.nb}
```

□ Printing

This sets the starting page number of the specified notebook to 15.

```
SetOptions[nb, StartingPageNumber → 15]
```

This sets the printing margins in the specified notebook. The option value is a list of four numbers, $\{l, r, b, t\}$, which specify the value of the left, right, bottom, and top margins in printer's points.

```
SetOptions[nb, PrintingMargins → {72,72,108,108}]
```

This modifies the notebook so the cell brackets are invisible when the notebook is printed.

```
SetOptions[nb, CellBrackets → False]
```

■ Palettes

□ Introduction

The *AuthorTools* palettes provide an easy point-and-click interface to the package functions. You can use the palettes to process either a single notebook or multiple notebooks in a single directory.

To process a single notebook:

1. Open the notebook. If it is already open, click the notebook to select it.
2. Click a button on the palette you want to use.

To process multiple notebooks:

1. Open the MakeProject dialog box.
2. Click Load and choose a project file. (If you have not already created a project file, see Creating a Project File to learn how.)
3. With the MakeProject dialog box as the input notebook, click a button on the palette you want to use. The palette functions will apply to all notebooks that belong to the specified project.

□ OpenAuthorTools

The OpenAuthorTools palette provides a convenient way to access the other palettes in the package. Choose any topic to open the corresponding palette. Click the top button of this or any other palette to get help.



- **MakeProject**—set up and handle projects involving multiple notebooks
- **MakeIndex**—create an index
- **MakeContents**—create a table of contents
- **MakeCategories**—create a browser categories file to customize the contents of the Help Browser
- **MakeBilateralCells**—create a bilateral cell to display example calculations
- **NotebookDiff**—find differences between notebooks
- **NotebookRestore**—retrieve content from a notebook that contains a syntax error
- **Paginate**—set page numbers
- **ExportCells**—extract all cells of a specified type, such as graphics cells
- **InsertValue**—insert the value of variables such as the current date and time
- **SetPrintingOptions**—specify properties of printed pages, such as headers and footers

□ **MakeProject**

Introduction

The MakeProject dialog box makes it easy to set up and manage projects involving multiple notebooks. For example, you can use this dialog box to generate a unified table of contents or index for a set of notebooks.



Creating a Project File

The first step in processing multiple notebooks is to create a project file. This is a file that specifies the names and location of all notebooks in a project. You can create any number of project files, one for each project you are working on. Each project file has a `.m` suffix.

To create a project file:

1. Open the MakeProject dialog box.
2. Type in a name for your project in the Name of Project text box.
3. Choose Target Directory. A file browse dialog box appears.
4. Select a file in the directory containing your source notebooks. The full pathname of the directory is pasted into the Directory text box.
5. Choose Select All. A list of all notebook files in the directory you specified appears in the Files text box.
6. Edit the list to delete any notebooks that do not belong in the project.
7. Choose Save under the Project File tab to save the project file.

Handling a Project

To process all notebooks in a project:

1. Open the MakeProject dialog box.
2. Click Load and choose the project file you need. (If you have not already created a project file, see Creating a Project File to learn how.)
3. With the MakeProject dialog box as the input notebook, click a button on the palette you want to use. The palette functions will apply to all notebooks in the specified project.

Once you have loaded a project, you can use the buttons under the Project Actions tab to perform a variety of tasks.

- To create a table of contents in a specific style, click the corresponding button under Contents.
- To create an index in a specific style, click the corresponding button under Index.
- To create a BrowserCategories.m file, choose BrowserCategories under Other.
- To create a BrowserIndex.nb file, choose BrowserIndex under Other.
- To create page numbers to all notebooks in the project, choose Paginate under Other.

□ **MakeContents**

Introduction

The MakeContents palette enables you to generate a table of contents for any notebook or group of notebooks.



You can format your table of contents in any of three predefined styles:

- Simple—lists all the entries without page numbers or any special formatting. A Simple table of contents is like an outline. It is useful for checking that your topics are in the right order before generating a table of contents in one of the other two styles.
- Book—resembles a table of contents in a typical book, with each topic and subtopic on a separate line.
- Condensed—combines multiple subtopics on the same line to achieve a more compact appearance.

Creating a Table of Contents

To create a table of contents:

1. For a single notebook, open the notebook and make it the currently selected notebook. For multiple notebooks, open the MakeProject dialog box and load the project file for your project. (If you have not already created a project file, see *Creating a Project File* to learn how.)
2. Open the MakeContents palette.
3. Choose **Paginate** to assign page numbers to the source notebooks. You can skip this step if you have paginated the notebooks previously or are creating a Simple table of contents, which does not contain any page numbers.
4. Choose either **Make Simple Contents**, **Make Book Contents**, or **Make Condensed Contents**, depending on which style of table of contents you want to create.

It may take a few moments for the evaluation to be completed, depending on the size of the source notebooks. The newly generated table of contents appears on the screen and is saved in the same directory as the source notebook.

Note: *Mathematica* automatically adds cell tags to the source notebooks and saves all changes. It is therefore advisable to make a back-up copy of your notebooks before generating a table of contents.

Customizing a Table of Contents

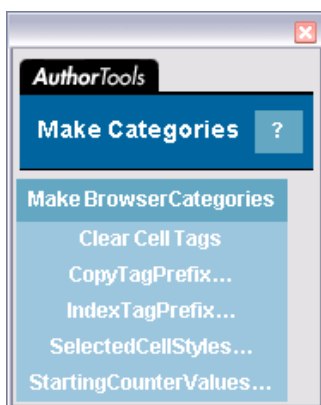
The buttons at the bottom of the palette let you specify options to customize various features of the table of contents. To set the value of an option, click the button bearing the name of that option. This brings up a dialog box displaying the current value of the option. You can edit the text and click Apply for the changes to take effect. Click OK to close the dialog box.

- **Pageinate** calculates page numbers for the source notebooks. The page numbers are saved as **TaggingRules** in the notebook. By default, the first notebook has starting page number 1.
- **Clear Cell Tags** removes the automatically generated cell tags in the source notebooks. If you remove the cell tags, the hyperlinks in the table of contents will take you to the start of the notebook instead of to a specific topic.
- **CellTagPrefix** lets you specify a prefix to the automatically generated cell tags.
- **ContentsFileName** lets you specify the name of the notebook containing the table of contents.
- **SelectedCellStyles** lets you specify the cell styles that should be included in the table of contents. The default setting is **SelectedCellStyles** → {"Title", "Section", "Subsection", "Subsubsection"}.

□ MakeCategories

Introduction

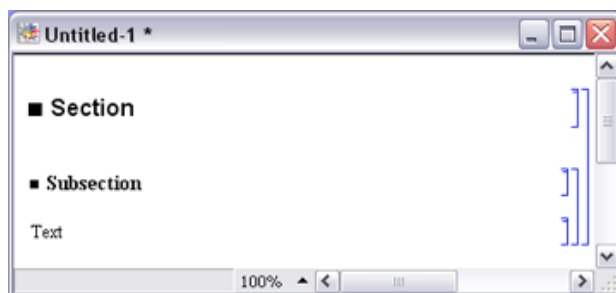
A **BrowserCategories.m** file determines how information is organized and displayed in the *Mathematica* Help Browser. The **MakeCategories** palette enables you to generate a **BrowserCategories.m** file for your notebook or project. By creating such a file, you can add your own content to the online help so it is accessible from the Help Browser.



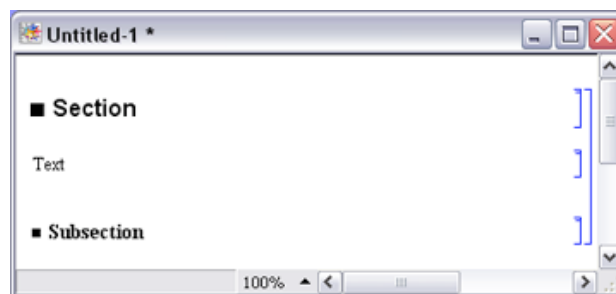
Outline Format

When generating a browser categories file, *Mathematica* assumes the source notebook is in strict outline format. This means all cells in the notebook are arranged in a consistent hierarchical order.

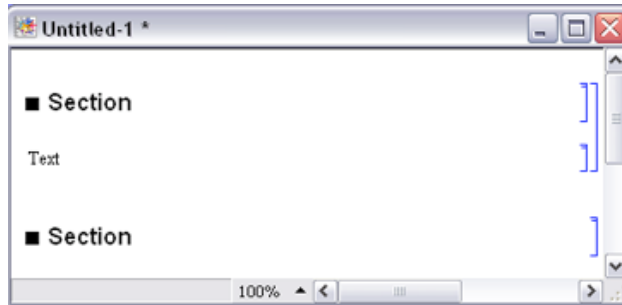
Consider a notebook composed entirely of Section, Subsection, and Text cells. If every Text cell is contained in a Subsection cell and every Subsection cell is contained in a Section cell, as shown here, the notebook is said to be in strict outline format.



However, if the following sequence of cells occurs anywhere in the notebook, the hierarchical structure breaks down. This is because in the resulting browser categories file, the Section cell will have to serve both as an item (to display the Text cell) and a category (to contain the Subsection cell), which is not possible.



Note that if a Section cell contains only Text cells and no Subsection cells as shown here, the strict outline format is again restored, since now each Section cell can serve as an item.



Note: If you attempt to generate a browser categories file from a notebook that is not in strict outline format, you will get a warning message. The resulting browser categories file will ignore the presence of any cells that stray from the strict outline format, so those cells will not be viewable in the Help Browser.

Creating Browser Categories

To generate a browser categories file:

1. For a single notebook, open the notebook and make it the currently selected notebook. For multiple notebooks, open the MakeProject dialog box and load the project file for your project. (If you have not already created a project file, see Creating a Project File to learn how.)
2. Open the MakeCategories palette.
3. Choose Make Browser Categories.

A browser categories file is created and saved in the same directory as your source notebook. Cell tags are automatically added to the input notebook.

Note: When you generate a browser categories file, your source notebooks are modified and all changes are automatically saved. It is therefore advisable to make a back-up copy of your notebook before using this palette.

The BrowserCategories palette automatically inserts cell tags into the target notebook. To remove the automatically generated cell tags in the source notebook(s), click Clear Cell Tags. If you remove the cell tags, the hyperlinks in the table of contents will take you to the start of the notebook instead of to a specific topic.

Customizing the Browser Categories

The three buttons at the bottom of the palette enable you to specify options to customize various features of the BrowserCategories.m file. To set the value of an option, click the button bearing the name of that option. This brings up a dialog box displaying the current value of the option. You can edit the text and click Apply for the changes to take effect. Click OK to close the dialog box.

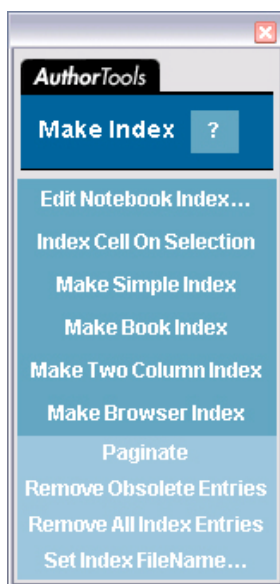
Here are the options:

- **CopyTagPrefix** lets you specify a string that is prepended to all copy tags in the `BrowserCategories.m` file. The default setting for this option is `CopyTagPrefix → "b: "`.
- **IndexTagPrefix** lets you specify a string that is prepended to all index tags in the `BrowserCategories.m` file. The purpose of this option is to make the index tags unique among all applications installed in the Help Browser. If set to `Automatic`, `IndexTagPrefix` defaults to the setting for `ProjectName`.
- **SelectedCellStyles** lets you specify the cell styles that are used in creating the browser categories file. Possible settings are `Automatic` or a list of cell styles. The default setting for this option is `SelectedCellStyles → {Title, Section, Subsection, Subsubsection}`.
- **StartingCounterValues** lets you specify the autonumbering of chapters, sections, subsections, and so on. A setting of `StartingCounterValues → {1,0,1,0}`, for example, will number the chapters starting at 1, the sections starting at 0, the subsections starting at 1, and so on. The default setting for the option, `StartingCounterValues → {0,0,0,0}`, numbers everything starting at 0.

□ **MakeIndex**

Introduction

The **MakeIndex** palette enables you to generate an index for your notebook or project. *Mathematica* calculates page numbers for all the index entries you specify and lists them in alphabetical order. Each entry in the index is hyperlinked to the related material in the source notebooks.



To create an index for a document, you must assign an index entry to each cell in the notebook that will be referenced in the index. This is done using the Edit Notebook Index dialog box of the MakeIndex palette.

Once all the index entries have been defined, you can use the palette to generate the index. For any document, you can generate several different types of indexes.

Simple Index

A Simple Index is used to check that all the index entries you specified were entered correctly. It lists all index entries, including duplicates and subentries, on separate lines. The text of the hyperlink for each entry consists of the cell tag of the target material and the name of the notebook in which it occurs.

The Simple Index is saved in the same directory as the source notebooks. By default the file is called SimpleIndex.nb, but you can specify a different name using the Set Index FileName button at the bottom of the palette.

Book Index

A Book Index contains all the index entries formatted with the following features.

- Duplicate index entries to the same page are removed. Duplicate index entries on consecutive pages in the same notebook are combined into a single page range.
- Subindex entries are formatted on a separate line and indented as shown here.
 element, 2
 Fuzzy Logic Pack, 1
 loading, 1-2
 Fuzzy Sets, 1, 3
 as functions, 1
 as models, 4
- The text of each hyperlink is the page number(s) on which the index entry appears.

If the Book index builds without error, it is saved in the same directory as the source notebooks. By default the file is called BookIndex.nb, but you can specify a different name using the Set Index FileName button at the bottom of the palette.

Two-Column Index

A two-column index has the same features as a Book Index, except that the entries are arranged in two columns.

If the Two-Column Index builds without error, it is saved in the same directory as the source notebooks. By default the file is called TwoColumnIndex.nb, but you can specify a different name using the Set Index FileName button at the bottom of the palette.

Browser Index

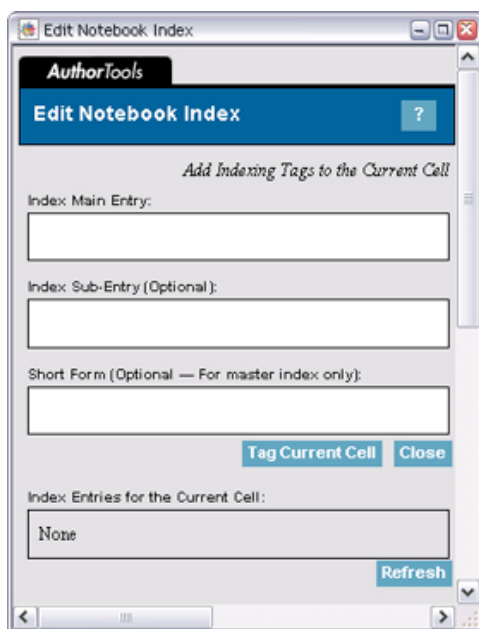
A browser index file defines a list of keywords to be included in the Master Index of the Help Browser. By creating a browser index for your notebooks, you can integrate your own reference material into the existing online help.

For the `BrowserIndex.nb` file to be compiled without error, an appropriate browser categories file must exist in the same directory as the source notebooks. To create a browser categories file, you can use the *MakeCategories* palette included with *AuthorTools*.

Setting Up an Index

Setting Up Index Entries

To create an index, you must associate index entries with specific cells in your source notebooks. This is done using the Edit Notebook Index dialog box, which allows you to add, edit, and remove index entries.



This dialog box contains three text boxes:

Index Main Entry—This specifies a term in the index.

Index Sub-Entry—This is useful for distinguishing two cells that fall under the same main entry.

Short Form—This defines a short word or phrase to represent the entry in the Master Index of the Help Browser.

To associate an index entry with a cell (or group of cells):

1. Select the cell(s).

2. Specify the entry by typing it into the text fields of the dialog box. The main entry and subentry can be either a string or a two-dimensional expression, but the short form must be a string.
3. Choose Tag Current Cell to assign the entry to the selected cell(s).

Any index entries associated with a cell are displayed in the box at the bottom of the dialog box, if you select that cell. After you add a new entry, you can choose Update Tag List to check that the entry you specified has been assigned.

Note: Alternatively, you can select all or part of a cell's contents and click the Index Cell on Selection button. This makes the index entry for the cell the same as the selected text.

To edit or remove an index entry:

1. Select the cell. A list of the currently assigned entries associated with that cell is displayed at the bottom of the dialog box. Each entry also has hyperlinks marked Edit and Remove next to it.
2. Click Edit to edit the entry or Remove to remove it.
3. Edit the entries displayed in the text fields of the dialog box.
4. Choose Tag Current Cell. This replaces the entry you edited with the new entry.

To generate an index:

1. For a single notebook, open the notebook and make it the currently selected notebook. For multiple notebooks, open the MakeProject dialog box and load the project file for your project. (If you have not already created a project file, see Creating a Project File to learn how.)
2. Open the MakeIndex palette.
3. Click Edit Notebook Index. This brings up a dialog box.
4. Use this dialog box to associate index entries with all cells in the notebook that are to be referenced in the index.
5. Choose Paginate to assign page numbers to the source notebooks. You can skip this step if you have paginated the notebooks previously or if you are creating a Simple index, which does not contain any page numbers.
6. Click the button corresponding to the type of index you want to create. For example, to build a Simple index choose Make Simple Index. The other formats available are Book Index, Two Column Index, and Browser Index. (See MakeIndex to learn about the differences among these four types of index.)

Note: *Mathematica* automatically creates an index based on all the index entries that you defined. The index is saved as a separate notebook in the same directory as the source notebook.

Browser Index

When you choose an entry in a browser index, the target cells are displayed in the Help Browser. For the index entries to work properly, an appropriate browser categories file must be present in the same directory as the source notebooks.

If you associate an index entry with a cell that cannot be displayed in the Help Browser, the index entry will bear the label “match not found” to indicate that the entry does not have a well-defined target.

For a cell to be displayed in the Help Browser, the browser categories file must contain an item corresponding to that cell. If no such item is present, the cells are not the target of a terminal category in the columns of the Help Browser and so the index entry cannot link to those cells.

This can happen if the source notebook is not in strict outline format and the target cells fall outside the outline. Even if the source notebook is in outline format but the target cell is a heading cell that does not correspond to an item in the browser categories file, the index entry will not work.

To fix an index entry that does not have a match:

1. Edit the browser categories file by hand to insert an item corresponding to the target cells for the index entry. (For details of how browser categories files work, see *Creating a Browser Categories File*.)
2. Use the MakeIndex palette to regenerate the browser index.

□ Paginate

Introduction

The Paginate palette enables you to assign page numbers to one or more notebooks. You must paginate your source notebooks at least once before you generate a table of contents or index. Otherwise, the table of contents or index will not contain any page numbers.



Paginating Notebooks

To paginate one or more notebooks:

1. For a single notebook, open the notebook and make it the currently selected notebook. For multiple notebooks, open the MakeProject dialog box and load the project file for your project. (If you have not already created a project file, see Creating a Project File to learn how.)
2. Open the Paginate palette.
3. Choose Paginate.

Options for Pagination

The three buttons at the bottom of the palette enable you to specify options that control how the page numbers are assigned. To set the value of an option, click the button bearing the name of that option. This brings up a dialog box displaying the current value of the option. You can edit the text and click Apply for the changes to take effect. Click OK to close the dialog box.

Here are the options:

- **StartingPages** determines the starting page number for all notebooks in a project. The option is specified as a list, $\{p_1, p_2, \dots, p_n\}$, where p_k specifies the starting page number of the k th notebook in the project. Each element of the list can be set to an integer, "Next", "Odd", "Even", or Inherited. The default setting is **StartingPages** $\rightarrow \{1, \text{"Next"}\}$.

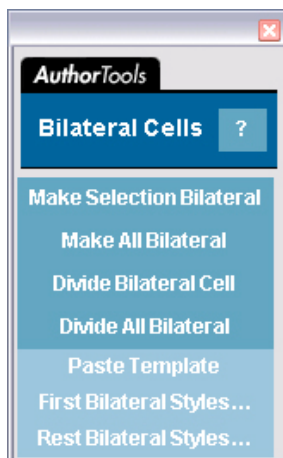
If p_k is "Next", "Odd", or "Even", the starting page for the k th notebook is the next page, next odd page, or next even page following the last page of the previous notebook. If p_k is Inherited, the starting page number for the k th notebook is inherited from the setting of the option **StartingPageNumber** for that notebook.

- **OpenAllCellGroups** determines whether to open all cell groups before calculating page breaks. The default setting is **OpenAllCellGroups** $\rightarrow$ True.
- **PaginationFunctions** specifies a function (*func*) to apply to each notebook after page numbers are calculated but before the notebook is closed. If this option is set to something other than Null, the return values from **Paginate** will have *func[nb]* appended to each sublist.

□ MakeBilateralCells

Introduction

The MakeBilateralCells palette allows you to display *Mathematica* calculations in a compact, bilateral format. Each bilateral cell consists of expository text to the left and input and output cells to the right. Bilateral cells are used extensively in *The Mathematica Book* to display examples.



Creating Bilateral Cells

To convert a *Mathematica* example into bilateral form:

1. Create the example you want to convert to bilateral form. By default, each example must consist of a single MathCaption cell followed by any number of input and output cells. However, you can change the cell styles used to construct a bilateral cell, as explained in Specifying Cell Styles.
2. Open the MakeBilateralCell palette.
3. Select the cells you want to format.
4. Choose Make Selection Bilateral.

The selected cells are automatically converted into bilateral cells. The text from each MathCaption cell appears in the left column while the corresponding input and output cells appear in the right column.

For example, consider the following sequence of cells:

Here is the integral $\int x^n dx$ in *Mathematica*.

Integrate[x^n, x]

$$\frac{x^{1+n}}{1+n}$$

If you select the cells and choose Make Selection Bilateral, all three cells are combined into a bilateral cell.

Here is the integral $\int x^n dx$ in *Mathematica*.

In[1]:= **Integrate[x^n, x]**
Out[1]= $\frac{x^{1+n}}{1+n}$

If you have a notebook containing multiple *Mathematica* examples, you can convert all the examples into bilateral form in one step. Each example must consist of a single MathCaption cell followed by any number of input and output cells.

To convert all examples in a notebook or project into bilateral form:

1. For a single notebook, open the notebook and make it the currently selected notebook. For multiple notebooks, open the MakeProject dialog box and load the project file for your project. (If you have not already created a project file, see Creating a Project File to learn how.)
2. Open the MakeBilateralCell palette.
3. Choose Make All Bilateral.

Mathematica automatically finds all cases in the notebook where input and output cells follow a MathCaption cell and converts each such example into a single bilateral cell.

Dividing Bilateral Cells

Once you have created a bilateral cell, you can reverse the process and convert the cell back into its constituent Input, Output, and MathCaption cells. To do this, select the bilateral cell and choose Divide Bilateral Cell. To divide all bilateral cells in a notebook in one step, select the notebook and choose Divide All Bilateral.

Updating Bilateral Cells

Once you have created a bilateral cell, you cannot evaluate its input cells. If you want to modify the input and re-evaluate, you must divide the bilateral cell, evaluate the raw input cells, and then change your cells back into bilateral form.

To modify input and output in a bilateral cell:

1. Select the bilateral cell and decompose it by choosing Divide Bilateral Cell.
2. Edit the cells you want to modify.
3. Select any Input cell you edited and re-evaluate it (by pressing SHIFT+ENTER). A new Output cell is generated.
4. Reformat the group of cells by selecting it and choosing Make Selection Bilateral.

Pasting Bilateral Templates

Another way to create a bilateral cell is to first create a template and then add the content you need in the appropriate text fields.

To create a template for a bilateral cell:

1. Open the MakeBilateralCell palette.

2. Place the cursor at the point where you want the bilateral cell to be created.
3. Choose Paste Template. A template is pasted into the notebook, containing some sample text.
4. Replace the sample text with material appropriate for your example.

Specifying Cell Styles

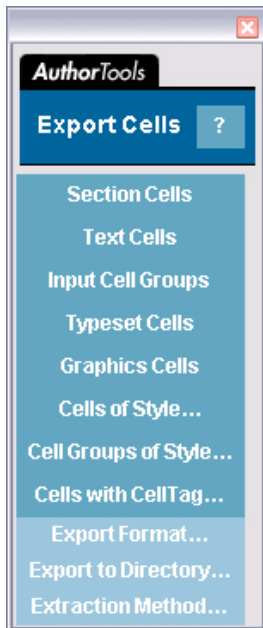
The two buttons at the bottom of the palette allow you to specify what cell styles should be used in creating a bilateral cell. To set the value of either option, click the appropriate button to bring up a dialog box. You can edit the text and click Apply for the changes to take effect. Click OK to close the dialog box.

- First Bilateral Styles specifies the cell styles that should be used for the caption of a bilateral cell. The default setting is `$FirstBilateral`Styles`. This typically has the value `{"MathCaption"}`.
- Rest Bilateral Styles determines what cell styles should be used for the content of a bilateral cell. The default setting is `$RestBilateral`Styles`. This typically includes Input, Output, and Graphics cells.

□ ExportCells

Introduction

The ExportCells palette enables you to extract content of a specified type from a given notebook or project and save the content in a desired format. For example, you can extract all the graphics cells from a notebook and save each graphic as a separate GIF file.



You can extract content of the following types:

- all cells of a particular style, such as all Section or Text cells
- all cell groups with a heading cell of a particular style, such as all input cell groups
- all cells with a specified cell tag

By default, the content is saved in a new notebook. However, you can choose from any of the other export formats supported by *Mathematica*: plain text, HTML, $\text{\TeX}$, GIF, JPEG, BMP, EPS, TIFF, PICT, and so on. To view a complete list of possible formats, choose `ExportFormat` at the bottom of the palette.

Exporting Cells

To export all cells of a specified style:

1. For a single notebook, open it and make it the currently selected notebook. For multiple notebooks, open the `MakeProject` dialog box and load the project file for your project. (If you have not already created a project file, see `Creating a Project File` to learn how.)
2. Open the `ExportCells` palette.
3. Choose `Cells of Style`.
4. In the dialog box that appears, enter the style of the cells you want to export.
5. Click `OK` to close the dialog box and apply the changes to your notebook. Click `Cancel` if you do not want the changes to be saved. Click `Apply` to apply the changes without closing the dialog box. Then click `OK`.

The cells you specified are extracted from the source notebooks and saved in the same directory. By default, the exported cells are saved in a new notebook. If you want to export cells in a different format, you must set the value of the option `ExportFormat` as explained in `Specifying the Export Format`.

To export all cell groups of a specified style:

1. For a single notebook, open the notebook and make it the currently selected notebook. For multiple notebooks, open the `MakeProject` dialog box and load the project file for your project.
2. Open the `ExportCells` palette.
3. Choose `Cell Groups of Style`.
4. In the dialog box that appears, enter the style of the heading cells for the cell groups you want to export. Then click `OK`.

The cells you specified are extracted from the source notebooks and saved in the same directory. By default, the exported cells are saved in a new notebook. If you

want to export cells in a different format, you must set the value of the option `ExportFormat`, as explained in Specifying the Export Format.

To export all cells having a specified cell tag:

1. For a single notebook, open the notebook and make it the currently selected notebook. For multiple notebooks, open the `MakeProject` dialog box and load the project file for your project.
2. Open the `ExportCells` palette.
3. Choose `Cells with CellTag`.
4. In the dialog box that appears, enter the cell tag of the cells you want to export. Then click `OK`.

All cells with the cell tag you specified are extracted from the source notebooks and saved in the same directory. By default, the exported cells are saved in a new notebook. To export cells in a different format, you must set the value of the option `ExportFormat` as explained in Specifying the Export Format.

Specifying the Export Format

To specify the export format:

1. Choose `ExportFormat`. This brings up a dialog box showing the current setting of the `ExportFormat` option.
2. Specify the format you want to save in by typing it in the text box. You can also click a button bearing the name of a specific format to paste that format into the text box.
3. Click `Apply` for the new setting to take effect. Click `OK` or `Close` to close the dialog box.

Specifying the Export Directory

To specify the directory to export to:

1. Choose `Export To Directory`. This brings up a dialog box showing the current setting of the `ExportDirectory` option.
2. In the text box, enter the full pathname of the directory to which you want to save. You can also use the `Browse` button to bring up a file dialog box, then locate a file in the target directory and click `Open`.
3. Click `Apply` for the option to take effect. Click `OK` to close the dialog box.

Specifying the Extraction Method

To specify the extraction method:

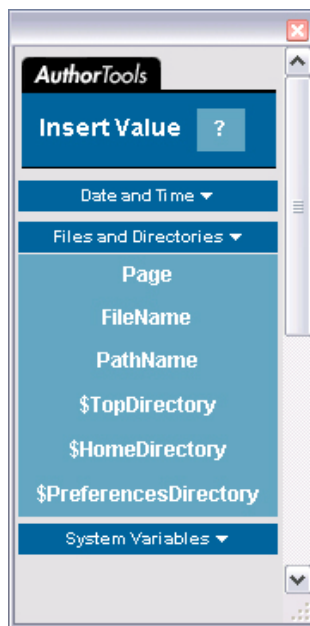
1. Choose `ExtractionMethod`. This brings up a dialog box showing the current setting of the `ExtractionMethod` option.
2. Enter the extraction method you want to use in the text box. Possible settings are:

- NotebookGet, which is faster but uses more memory
 - NotebookRead, which is slower but uses less memory
3. Click Apply for the new setting to take effect. Click OK or Close to close the dialog box.

□ InsertValue

Introduction

Using the InsertValue palette, you can insert an object to display the current value of variables such as the names of the home directory or the preferences directory, as well as the current filename, pathname, date, or time. The display object, once inserted, is dynamically updated so it always reflects the current value of the variable.



For example, the text in the following cell contains display objects that refer to the current date and time. This ensures that each time the notebook is opened, the cell always shows the current date and time.

Today's date is 5/11/04.
The current time is 10:04:10.

You can choose from variables divided into three categories.

- Date and Time—Enter the current date or time in various formats.
- Files and Directories—Enter the current filename, pathname, page number, or the values of selected directories.

- **System Variables**—Enter system parameters, such as the operating system or version number of *Mathematica*.

Inserting a Value

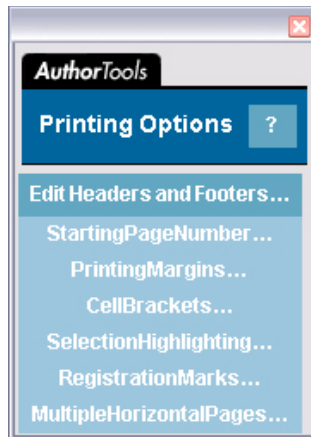
To insert the value of a variable:

1. Open the InsertValue palette.
2. Choose any of the three categories: Date and Time, Files and Directories, or System Variables to view the buttons in that category.
3. Click the button with the name of the variable whose value you want to enter. The value of the selected variable is pasted into the notebook at the position of the cursor.

□ SetPrintingOptions

Introduction

The SetPrintingOptions palette allows you to set options that control how a notebook is printed. You can use this palette, for example, to specify headers and footers, set the starting page number, or control the size of margins.



Headers and Footers

You can specify different headers/footers for left and right pages. In addition, for each left or right page, you can specify three different types of headers/footers: left-aligned, right-aligned, and centered.

To specify headers/footers for a document:

1. Open the SetPrintingOptions palette.
2. Choose Edit Headers and Footers. This brings up a dialog box.
3. Choose one of the horizontal tabs bearing the name of the header/footer you want to specify. A set of text boxes is displayed.

4. Enter the header/footer you want in the appropriate text box. You can either type text directly or choose **Insert Value**. This opens the **InsertVariables** palette for entering variables such as the filename or date.
5. To enter a running head, choose **Insert Running Head**, type a cell style in the dialog box that appears, and click **OK**.
6. When you have finished editing, click **OK** to close the **Headers and Footers** dialog box and apply the changes to your notebook. Click **Cancel** if you do not want the changes to be saved.

To specify the facing of pages:

Choose the appropriate setting for the **Set Facing Pages** option at the top of the dialog box. This option can be set to one of three different settings.

- **First page facing left**—This is indicated by two boxes to the right of the button with the number 1 in the left box.
- **First page facing right**—This is indicated by two boxes to the right of the button with the number 1 in the right box.
- **Left and right pages are equivalent**—This is indicated by a single box to the right of the button with the number 1 inside it.

For the first two cases, you can specify separate headers/footers for left and right pages. For the third case, you can specify headers/footers for one type of page only. The number of text boxes available for displaying headers/footers change automatically to reflect this difference.

To control whether headers/footers are printed on the first page:

Click one of the radio buttons marked **Yes** or **No** after the “Header/Footer on First Page?” statement.

To insert a line below/above all headers/footers:

Click one of the radio buttons marked **Yes** or **No** after the “Left/Right Header/-Footer Lines?” statement.

Starting Page Number

To specify the starting page number for the printed notebooks:

1. For a single notebook, open the notebook and make it the currently selected notebook. For multiple notebooks, open the **MakeProject** dialog box and load the project file for your project. (If you have not already created a project file, see **Creating a Project File** to learn how.)
2. Open the **SetPrintingOptions** palette.
3. Choose **StartingPageNumber**. This brings up a dialog box.
4. Enter the starting page number.
5. When you have finished editing, click **OK** to close the dialog box and apply the changes to your notebook. Click **Cancel** if you do not want the changes to be saved. Click **Apply** to apply the changes without closing the dialog box.

Printing Margins

To set the margins for the printed notebooks:

1. For a single notebook, open the notebook and make it the currently selected notebook. For multiple notebooks, open the MakeProject dialog box and load the project file for your project. (If you have not already created a project file, see Creating a Project File to learn how.)
2. Open the SetPrintingOptions palette.
3. Choose PrintingMargins. This brings up a dialog box.
4. Enter the left, right, bottom, and top margins in printer's points (72 points equal one inch).
5. When you have finished editing, click OK to close the dialog box and apply the changes to your notebook. Click Cancel if you do not want the changes to be saved. Click Apply to apply the changes without closing the dialog box.

Cell Brackets

To specify whether cell brackets should be printed:

1. For a single notebook, open the notebook and make it the currently selected notebook. For multiple notebooks, open the MakeProject dialog box and load the project file for your project. (If you have not already created a project file, see Creating a Project File to learn how.)
2. Open the SetPrintingOptions palette.
3. Choose PrintCellBrackets. This brings up an editing dialog box.
4. Click True if you want cell brackets to be printed and False otherwise.
5. When you have finished editing, click OK to close the dialog box and apply the changes to your notebook. Click Cancel if you do not want the changes to be saved. Click Apply to apply the changes without closing the dialog box.

Selection Highlighting

To specify whether the current selection should appear highlighted when a notebook is printed:

1. For a single notebook, open the notebook and make it the currently selected notebook. For multiple notebooks, open the MakeProject dialog box and load the project file for your project.
2. Open the SetPrintingOptions palette.
3. Choose SelectionHighlighting. This brings up an editing dialog box.
4. Click True if you want the notebook selection to appear highlighted in print and False otherwise.
5. When you have finished editing, click OK to close the dialog box and apply the changes to your notebook. Click Cancel if you do not want the

changes to be saved. Click **Apply** to apply the changes without closing the dialog box.

Registration Marks

To specify whether trim marks should be added to indicate the corners of a printed page:

1. For a single notebook, open the notebook and make it the currently selected notebook. For multiple notebooks, open the **MakeProject** dialog box and load the project file for your project.
2. Open the **SetPrintingOptions** palette.
3. Choose **RegistrationMarks**. This brings up an editing dialog box.
4. Click **True** if you want trim marks to be printed and **False** otherwise.
5. When you have finished editing, click **OK** to close the dialog box and apply the changes to your notebook. Click **Cancel** if you do not want the changes to be saved. Click **Apply** to apply the changes without closing the dialog box.

Multiple Horizontal Pages

To specify whether cells that extend beyond the width of the page should be printed on additional pages:

1. For a single notebook, open the notebook and make it the currently selected notebook. For multiple notebooks, open the **MakeProject** dialog box and load the project file for your project. (If you have not already created a project file, see **Creating a Project File** to learn how.)
2. Open the **SetPrintingOptions** palette.
3. Choose **MultipleHorizontalPages**. This brings up an editing dialog box.
4. Click **True** if you want the cells to continue printing on another page. Click **False** if you want the cells to be chopped off at the edge of the page.
5. When you have finished editing, click **OK** to close the dialog box and apply the changes to your notebook. Click **Cancel** if you do not want the changes to be saved. Click **Apply** to apply the changes without closing the dialog box.

About the Author

Pavi Sandhu has a Ph.D. in physics from Boston University. He is the author of *The MathML Handbook*, published by Charles River Media in November 2002.

Pavi Sandhu

Senior Technical Writer
Oracle Corporation
Redwood Shores, CA

Integral Equations

Stan Richardson

We show how *Mathematica* can be used to obtain numerical solutions of integral equations by exploiting a combination of iteration and interpolation. The efficacy of the method is demonstrated by considering three classical integral equations of applied mathematics: Love's equation for the condenser problem, Theodorsen's equation associated with conformal mapping, and Nekrasov's equation arising in the theory of water waves. The success of the approach depends on the use of an appropriate method for the interpolation.

■ Introductory Example

As a first example, we consider the integral equation discussed by Kress [1, 2], namely

$$f(x) = e^{-x} - \frac{1}{2} + \frac{1}{2} e^{-(x+1)} + \frac{1}{2} \int_0^1 (x+1) e^{-xy} f(y) dy,$$

which has the exact solution $f(x) = e^{-x}$. Kress quotes errors only at the points $x = 0, 0.25, 0.5, 0.75$, and 1 , that at $x = 1$ always being the largest. Among the numerical methods and computations he describes for this example, this largest error is least for Nyström's method using Simpson's rule with 16 intervals, when it is 8.6×10^{-7} .

A classical method of solving such problems is to use iteration. Beginning with some suitable initial function $f_0(x)$, we construct a sequence of functions by

$$f_{n+1}(x) = e^{-x} - \frac{1}{2} + \frac{1}{2} e^{-(x+1)} + \frac{1}{2} \int_0^1 (x+1) e^{-xy} f_n(y) dy.$$

For this example, the iterative scheme converges for any continuous $f_0(x)$, and we choose to take $f_0(x) = 1$, setting this as

```
In[1]:= approxsoln[x_] = 1;
```

For comparison purposes, we also have the exact solution

```
In[2]:= exactsoln[x_] = e^-x;
```

and can keep track of the error using

```
In[3]:= error[x_] := exactsoln[x] - approxsoln[x]
```

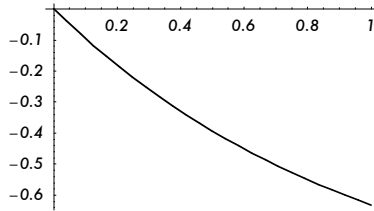
It is useful to have a visual record of this error, so we introduce

```
In[4]:= errorplot := Plot[error[x], {x, 0, 1}, PlotRange -> All]
```

At this stage, our approximate solution is not very impressive.

```
In[5]:= errorplot
```

From In[5]:=



With this example, it is theoretically possible to obtain analytic expressions for the functions $f_n(x)$, but, as this is not feasible for more general examples, we use a numerical approach and adopt the following procedure. With $f_n(x)$ known, we compute $f_{n+1}(x)$ from the iteration formula at a number of points in the interval $(0, 1)$, and make $f_{n+1}(x)$ an `InterpolatingPolynomial` fitted through these points. As a first try, we use a uniform distribution of points on $(0, 1)$, and take just 11 such points. The table of values is given by

```
In[6]:= values = Table[{x, e^-x - 1/2 + e^-x-1/2 + 1/2
NIntegrate[(x+1) e^-x y approxsoln[y], {y, 0, 1}]}], {x, 0, 1, 1/10}];
```

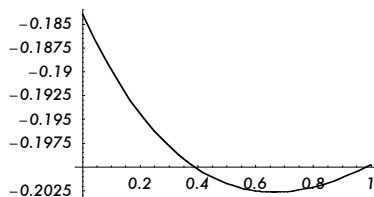
and the improved approximation is then

```
In[7]:= approxsoln[x_] = InterpolatingPolynomial[values, x];
```

As our plot shows, we now have a better approximation.

```
In[8]:= errorplot
```

From In[8]:=



We can combine all this into one step.

```
In[9]:= iterstep := (values =
Table[{x, e^-x - 1/2 + e^-x-1/2 + 1/2 NIntegrate[(x+1) e^-x y approxsoln[y],
{y, 0, 1}]}], {x, 0, 1, 1/10}];
approxsoln[x_] = InterpolatingPolynomial[values, x])
```

Now a little experimentation with `iterstep` and `errorplot` allows us to observe the convergence, but, once done, we know a suitable number of steps to take to obtain a satisfactory answer. For this first example, we demonstrate the convergence graphically in order to illustrate a number of features of our method. To accomplish this, we introduce `errorvalue`, which estimates the maximum error at each iteration from the adaptively-determined `PlotRange` of `errorplot`.

```
In[10]:= errorvalue := Max[Abs[
      Block[{$DisplayFunction = Identity}, Last[PlotRange[errorplot]]]]]
```

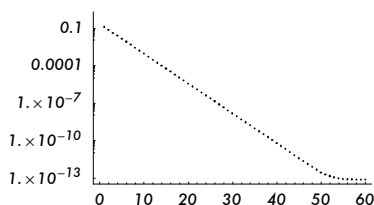
The convergence is nicely illustrated using `LogListPlot`, for which we need to load a package. In fact, other functions from the `Graphics`` package will be required later, so we use

```
In[11]:= << Graphics`
```

We now perform 60 further iterations and plot the result.

```
In[12]:= errorlist = Table[iterstep; errorvalue, {60}]; LogListPlot[errorlist]
```

From In[12]:=

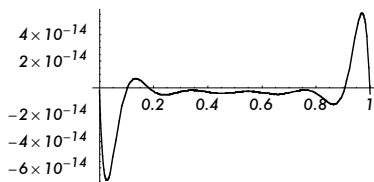


It is clear that we have now converged to an acceptable solution and that further iterations will produce no significant improvement. The remaining error is due solely to the interpolation process we have adopted: we have the best solution to our problem that can be represented in this form. As this makes clear, it is important that an appropriate interpolation routine be used. It is an instructive exercise to try using `Interpolation` with the present example, for this produces results that are vastly inferior, and later examples will require the use of more ingenuity in this respect.

We can show the distribution of the error in the solution we have obtained.

```
In[13]:= errorplot
```

From In[13]:=



When comparing with the results recorded by Kress, remember that he quotes errors only at five specific points.

The error distribution here is typical of that obtained when using a uniform distribution of interpolation points: the maximum errors occur near the ends of the interval. This feature can be eliminated and the maximum error reduced by using a nonuniform distribution of interpolation points that places more of them near the ends. Here, we can achieve this via a nonlinear scaling of the x -axis.

```
In[14]:= new[x_] := Sin[ $\frac{\pi x}{2}$ ]2
```

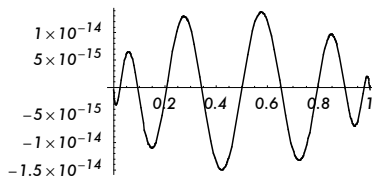
Modifying `iterstep` to incorporate this, we now use

```
In[15]:= iterstep :=  
  (values = Table[{new[x],  $e^{-\text{new}[x]} - \frac{1}{2} + \frac{1}{2} e^{-\text{new}[x]-1} + \frac{1}{2} \text{NIntegrate}[(\text{new}[x] + 1) e^{-\text{new}[x]} y \text{ approxsoln}[y], \{y, 0, 1\}]$ }, {x, 0, 1,  $\frac{1}{10}$ }}];  
  approxsoln[x_] = InterpolatingPolynomial[values, x])
```

and find that a few more iterations have indeed improved matters.

```
In[16]:= Do[iterstep, {5}]; errorplot
```

From In[16]:=



■ The Condenser Problem

An example discussed by Fox and Goodwin [3] in an early paper concerned with the numerical solution of integral equations is the equation derived by Love [4] to describe the electrostatic field produced by a condenser consisting of two parallel circular plates. This is

$$f(x) = 1 + \frac{1}{\pi} \int_{-1}^1 \frac{\kappa}{\kappa^2 + (x - y)^2} f(y) dy$$

when dimensionless variables are taken so that the plates have unit radius. Here κ is the distance between the plates. This is the relevant equation when the potentials of the plates are equal in magnitude but opposite in sign; if these are equal in both magnitude and sign, the first $+$ sign on the right-hand side becomes a $-$ sign.

Of particular interest in this problem is the *capacitance* of the condenser, which is the total charge on one of the plates, given in this nondimensional formulation by

$$\text{capacitance} = \frac{1}{\pi} \int_{-1}^1 f(x) dx.$$

Beginning again with

```
In[17]:= approxsoln[x_] = 1;
```

we now use

```
In[18]:= new[x_] := Sin[ $\frac{\pi x}{2}$ ]
```

and

```
In[19]:= iterstep :=
  (values = Table[{new[x], 1 +  $\frac{1}{\pi}$  NIntegrate[ $\frac{x \text{ approxsoln}[y]}{x^2 + (\text{new}[x] - y)^2}$ , {y,
    -1, new[x], 1}, MaxRecursion -> 20]}], {x, 0, 1,  $\frac{1}{6}$ }}];
  temp[x_] = InterpolatingPolynomial[Union[values,
    (values /. {x_?NumericQ, y_?NumericQ} -> {-x, y})], x];
  approxsoln[x_] =  $\frac{1}{2}$  (temp[x] + temp[-x]))
```

Here we have taken advantage of the fact that the solution $f(x)$ is an even function of x , so we need not compute entries in `values` for $x < 0$, and used a construction that ensures that our approximation is also an even function. There has also been a modest increase in the degree of the `InterpolatingPolynomial` employed.

We will judge convergence by the estimate we obtain for the capacitance. In fact, most numerical work actually quotes values for the capacitance multiplied by the factor $\pi/2$. This is convenient because a circular disc of unit radius on its own has a capacitance of $2/\pi$, so the difference between the value of this quantity and 1 is a measure of the interference effect between the two discs. This scaled capacitance tends to 1 as $\kappa \rightarrow \infty$. We therefore introduce

```
In[20]:= scaledcap :=  $\frac{1}{2}$  NIntegrate[approxsoln[x], {x, -1, 1}]
```

Fox and Goodwin [3] carried out their calculations for $\kappa = 1$, and we will begin with this value, too.

```
In[21]:=  $\kappa$  = 1;
```

A little experimentation with both the interpolation used and the number of iterations performed now tells us that we have convergence to 6 significant figures with

```
In[22]:= Do[iterstep, {20}]; scaledcap
```

```
Out[22]= 1.82078
```

Fox and Goodwin [3] do not evaluate the capacitance, but this was calculated for $\kappa = 1$, both from the figures given by Fox and Goodwin and independently, by Cooke [5], who gives the value 1.8208 for the scaled capacitance.

With this solution in hand, we can plot the equipotentials round the condenser. This was done manually by Love [6], but *Mathematica* makes the task somewhat easier. With cylindrical polar coordinates (r, θ, z) chosen so that a plate at potential +1 lies in the plane $z = +\kappa/2$ and one at potential -1 is in the plane $z = -\kappa/2$, both in the region $r < 1$, the potential V is independent of θ and is given by

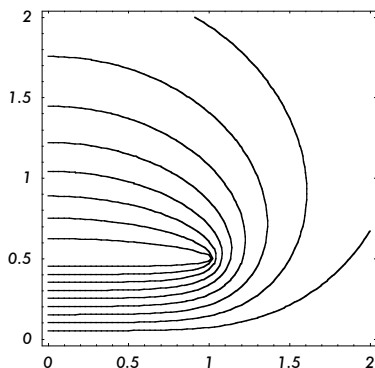
```
In[23]:= V[r_, z_] :=
  Which[z == 0, 0, z ==  $\frac{\kappa}{2}$  && Abs[r] ≤ 1, 1, z ==  $-\frac{\kappa}{2}$  && Abs[r] ≤ 1, -1, True,
  Chop[ $\frac{1}{\pi}$  NIntegrate[ $\left( \frac{1}{\sqrt{r^2 + (z - \frac{\kappa}{2} + i x)^2}} - \frac{1}{\sqrt{r^2 + (z + \frac{\kappa}{2} + i x)^2}} \right)$ 
    approxsoln[x], {x, -1, 1}, MaxRecursion → 20]]]
```

In fact, with `approxsoln` as a polynomial, *Mathematica* can perform the integrations here analytically, but a numerical approach seems preferable. We have also incorporated the known values of the potential on the plates and the plane $z = 0$ into this definition.

We can economize on the effort required to produce a reasonable sketch by exploiting the symmetry and plotting initially only in the first quadrant, where $r > 0$ and $z > 0$. We should also choose `PlotPoints` to ensure that `ContourPlot` evaluates the potential at the point on the edge of the disk.

```
In[24]:= contplot = ContourPlot[Evaluate[V[r, z]],
  {r, 0, 2}, {z, 0, 2}, ContourShading → False,
  Contours → Range[0.1, 0.9, 0.1], PlotPoints → 61]
```

From In[24]:=



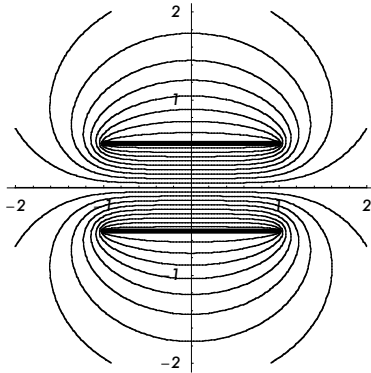
Now extract the information we need from this, and form the data required for the curves in the other quadrants.

```
In[25]:= lines1 = Cases[Graphics[contplot], _Line, ∞];
lines2 = lines1 /. {x_?NumericQ, y_?NumericQ} → {-x, y};
lines3 = lines1 /. {x_?NumericQ, y_?NumericQ} → {-x, -y};
lines4 = lines1 /. {x_?NumericQ, y_?NumericQ} → {x, -y};
```

Finally, we can Show all this, together with the lines omitted from contplot.

```
In[29]:= Show[Graphics[{lines1, lines2, lines3,
  lines4, Line[{{-2, 0}, {2, 0}}], AbsoluteThickness[2],
  Line[{{-1,  $\frac{\kappa}{2}$ }, {1,  $\frac{\kappa}{2}$ }}], Line[{{-1,  $-\frac{\kappa}{2}$ }, {1,  $-\frac{\kappa}{2}$ }}]}],
  AspectRatio → Automatic, Axes → True]
```

From In[29]:=



Whatever method one uses to solve the integral equation of this example, the computations become more critical as κ decreases. Cooke [5] takes his computations down to $\kappa = 0.4$, but records a value for $\kappa = 0.1$ obtained by Fox and Blake using the methods of Fox and Goodwin [3]; for this value of κ they find a scaled capacitance of 9.233. Using our approach, we need more iterations, but otherwise there are no problems.

```
In[30]:=  $\kappa = 0.1$ ; Do[iterstep, {150}]; scaledcap
```

```
Out[30]= 9.23285
```

Since this value is within 0.01 percent of that given by the asymptotic formula, valid for $\kappa \rightarrow 0$, obtained by Chew and Kong [7], namely

$$\text{In[31]:= } \text{approxscaledcap}[\kappa] := \frac{\pi}{4\kappa} + \frac{1}{4} \left(\text{Log}\left[\frac{16\pi}{\kappa}\right] - 1 \right) + \frac{\kappa}{16\pi} \left(\text{Log}\left[\frac{16\pi}{\kappa}\right]^2 - 2 \right)$$

```
In[32]:= approxscaledcap[0.1]
```

```
Out[32]= 9.23194
```

there is little practical need to carry out the computations with the integral equation for smaller values of κ . (See Chew and Kong's paper and the references therein for an account of the history of this formula; it is the product of over a century of effort by a large number of mathematicians.) However, *Mathematica* will do this if required. It is obviously sensible to reorganize the computations

presented in this rather discursive fashion into a more efficient program, but the main changes required are in the interpolation. For nonsmall κ , the graph of $f(x)$ is close to a parabola and `InterpolatingPolynomial` is appropriate, but as κ decreases the gradients become large near the ends of the interval $(-1, 1)$ and interpolation by neither polynomials nor splines is appropriate; `Fit` with suitable basis functions is more effective. Since this modification will be employed later in a different example, we will not illustrate it here.

It should be noted that the familiar “rule of thumb” criteria for stopping a sequence of iterations (e.g., continue iterating until the difference between two consecutive iterates is less than some predetermined small value) become more and more inadequate as κ decreases, because the operator involved approaches the identity as $\kappa \rightarrow 0$. This feature is independent of the means we use to represent our solution: the effective application of our method in dealing with nontrivial problems depends on some judicious experimentation with the method of interpolation employed and the number of iterations used.

Fabrikant [8] tabulates many values for the scaled capacitance, down to $\kappa = 0.001$, using a different method based on integral equations that produces upper and lower bounds. However, an asymptotic approach yields results of much greater accuracy for such values of κ . Fabrikant also quotes an error of approximately 0.01 percent for $\kappa = 0.1$, but his error estimates inevitably increase as κ decreases, while the asymptotic formula becomes *more* accurate for smaller values of κ .

■ Conformal Mapping

Many different methods exploiting integral equations have been developed to determine numerically the function giving the conformal map of the unit disk onto some given simply-connected domain; for an account of many of these, see Henrici [9]. As an example of an integral equation that is both nonlinear and singular, we consider the one introduced by Theodorsen for this purpose.

Let the unit disk reside in the plane of $z = r e^{i\theta}$ so its boundary is given by $r = 1$, and let the domain onto which we wish to map this disk reside in the plane of $\zeta = \rho e^{i\phi}$. We assume that the origin lies within this domain and that its boundary is given in polar form by some known function $\rho = \rho(\phi)$. The map we seek takes points on the boundary of the disk in the z -plane (identified by the angle θ) onto points on the boundary of the domain in the ζ -plane (identified by the angle ϕ). The problem reduces to finding the dependence between these two angles; once we have determined the function $\phi = \phi(\theta)$, the mapping itself can easily be constructed.

If we normalize the map by requiring that the origins in the two planes correspond, and that the map induces no rotation at the origin, the function $\phi(\theta)$ satisfies Theodorsen’s integral equation

$$\phi(\theta) = \theta + \frac{1}{2\pi} \int_0^{2\pi} \log(\rho(\phi(\tau))) \cot\left(\frac{\theta - \tau}{2}\right) d\tau.$$

The integral here is to be interpreted as a Cauchy principal value, there being a singularity at $\tau = \theta$. (It is possible to transform this equation to a form that is technically nonsingular (see Kythe [10], for example), but the removable singularity that results involves the computation of values as the ratio of small quantities, even if one manages to avoid $0/0$. It is generally preferable to keep the singularity and deal with it correctly, rather than disguise it and hope for the best.) We solve this equation using the iteration

$$\phi_{n+1}(\theta) = \theta + \frac{1}{2\pi} \int_0^{2\pi} \log(\rho(\phi_n(\tau))) \cot\left(\frac{\theta - \tau}{2}\right) d\tau.$$

Since an increase of 2π in θ must yield an increase of 2π in ϕ , a sensible approximate solution with which to begin the iteration is

```
In[33]:= approxsoln[θ_] = θ;
```

This is also the exact solution if, and only if, the domain in the ζ -plane is a circular disk centered on the origin.

As a first example, we will consider a domain bounded by a lemniscate of Booth (called an inverted ellipse by some authors because it is the inverse of an ellipse with respect to a circle), for which we know that the exact mapping function $\zeta = f(z)$ is given by

$$\text{In[34]:= } f[z_]:= \frac{2 p z}{1 + p + (1 - p) z^2}$$

Here p is a parameter satisfying $0 < p \leq 1$. For $p = 1$ the domain is a circular disk of unit radius, while as $p \rightarrow 0$ the domain degenerates to two circular discs of radius $1/2$ touching tangentially. We will consider the particular case

```
In[35]:= p = 0.6;
```

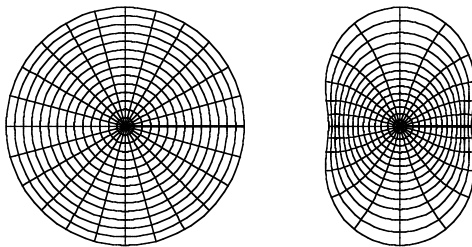
We can see how the mapping distorts the unit disk using `PolarMap` from the `Graphics`ComplexMap`` package that is already available to us.

```
In[36]:= originaldisk = PolarMap[Identity, {0, 1}, {0, 2 π}, Lines → {15, 25},
      PlotRange → All, Axes → False, DisplayFunction → Identity];
```

```
In[37]:= distortion = PolarMap[f, {0, 1}, {0, 2 π}, Lines → {15, 25},
      PlotRange → All, Axes → False, DisplayFunction → Identity];
```

```
In[38]:= DisplayTogetherArray[{originaldisk, distortion}]
```

From In[38]:=



We see that we have nonnegligible distortion, for the image domain is not convex. For this domain the function $\rho(\phi)$ is

$$\text{In[39]:= } \rho[\phi_]:= \sqrt{1 - (1 - p^2) \cos[\phi]^2}$$

and we can calculate that the exact solution $\phi(\theta)$ is

$$\text{In[40]:= } \text{exactsoln}[\theta_]:= \text{ArcTan}[p \text{ Tan}[\theta]]$$

for $-\pi/2 < \theta < \pi/2$; the restriction on the range here is necessary because it is not the principal value of the inverse tangent function that is relevant outside this range. (Using the two-variable form $\text{ArcTan}[\cos[\theta], p \sin[\theta]]$ instead gives a formula that is correct for $-\pi < \theta < \pi$, but the symmetry makes such an extension irrelevant here.) As before, we can introduce a plot to enable us to keep track of the error.

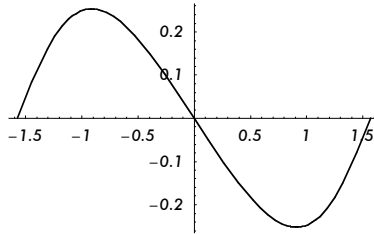
$$\text{In[41]:= } \text{errorplot} :=$$

$$\text{Plot}[\text{exactsoln}[\theta] - \text{approxsoln}[\theta], \{\theta, -\frac{\pi}{2}, \frac{\pi}{2}\}, \text{PlotRange} \rightarrow \text{All}]$$

Our initial approximation is not very accurate.

$$\text{In[42]:= } \text{errorplot}$$

From In[42]:=



We need another package to deal with the Cauchy principal value.

$$\text{In[43]:= } \ll \text{NumericalMath`CauchyPrincipalValue`}$$

Because the domain onto which we are mapping is symmetric about both axes in the ζ -plane, the function $\phi(\theta)$ must be the sum of θ and a function that is an odd function of both θ and $\theta - \pi/2$. It is natural to approximate this function using a linear sum of functions of the form $\sin(n\theta)$, where n must be an even integer. For this we use `Fit` with just 10 such functions, and therefore introduce

$$\text{In[44]:= } \text{basis} = \text{Table}[\text{Sin}[2 \text{ i } \theta], \{\text{i}, 10\}]$$

$$\text{Out[44]= } \{\text{Sin}[2\theta], \text{Sin}[4\theta], \text{Sin}[6\theta], \text{Sin}[8\theta], \text{Sin}[10\theta], \\ \text{Sin}[12\theta], \text{Sin}[14\theta], \text{Sin}[16\theta], \text{Sin}[18\theta], \text{Sin}[20\theta]\}$$

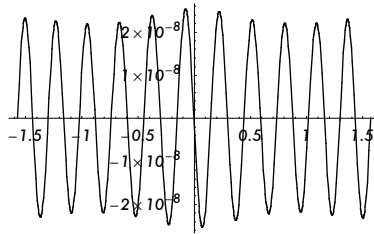
With this basis enforcing the correct symmetry, we need to compute values for the interpolation of `approxsoln` only for $0 < \theta < \pi/2$, and should not compute for $\theta = 0$ or $\theta = \pi/2$ because the symmetry already implies that we will have the correct values there. Such thinking leads us to define

```
In[45]:= iterstep :=
  (values = Table[{θ, 1/(2 π) CauchyPrincipalValue[Log[ρ[approxsoln[τ]]]
    Cot[θ/2]}, {τ, 0, {θ}, 2 π}], {θ, π/22, π/2 - π/22, π/22}];
  approxsoln[θ_] = θ + Fit[values, basis, θ])
```

Some preliminary trials now establish that we obtain satisfactory results with

```
In[46]:= Do[iterstep, {30}]; errorplot
```

From In[46]:=



At points in $|z| < 1$, the mapping function $f(z)$ is expressed in terms of $\phi(\theta)$ via the Cauchy integral formula, while on $|z| = 1$ the relationship between $f(z)$ and $\phi(\theta)$ is more immediate. As an approximation to $f(z)$ for $|z| \leq 1$, we have

```
In[47]:= approxf[z_] := Chop[If[Abs[z] == 1,
  ρ[approxsoln[Arg[z]]] Exp[i approxsoln[Arg[z]]], z Exp[1/(2 π)
  NIntegrate[(Log[ρ[approxsoln[θ]]] + i (approxsoln[θ] - θ)) /
    (1 - z Exp[-i θ]), {θ, -π, Arg[z], π}]]]]
```

$\text{Arg}[z]$ has been inserted into the range of integration here to cope with instances when $|z|$ is near 1 and the pole of the Cauchy integral is close to the contour of integration; *Mathematica* appreciates being told where problems are likely to occur!

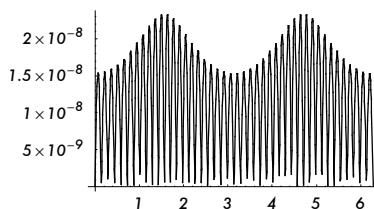
We can test the error at various points using

```
In[48]:= error[z_] := f[z] - approxf[z]
```

We can now plot the absolute value of the error on the boundary to assess the accuracy of our approximation there.

```
In[49]:= Plot[Abs[error[e^i θ]], {θ, 0, 2 π}]
```

From In[49]:=



We can also obtain an estimate of the overall accuracy in the interior by finding the maximum absolute value of the error at a random distribution of interior points. Because of the maximum modulus principle we, of course, expect the maximum error to occur on the boundary.

```
In[50]:= Table[error[Random[] ei Random[Real, {0, 2 π}]], {100}] // Abs // Max
```

```
Out[50]= 8.69614 × 10-9
```

We give a second example that requires more ingenuity; the difficulties will mean that *Mathematica* takes a little longer, but it can still cope. Consider the square in the plane of $\zeta = \xi + i\eta$ defined by $|\xi| < 1$ and $|\eta| < 1$. The exact mapping function is given by a Schwarz-Christoffel transformation and can be expressed in terms of elliptic integrals (see Kober [11], for example), but *Mathematica* handles the computations more efficiently if we use numerical integration and define the mapping function as follows.

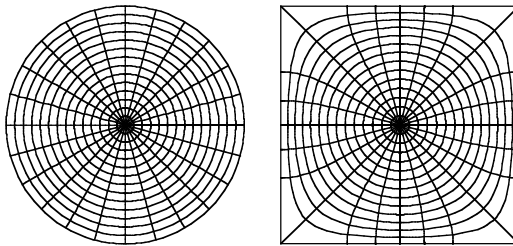
```
In[51]:= Clear[f]; f[z_?NumericQ] := Chop[
$$\frac{2 NIntegrate\left[\frac{1}{\sqrt{1+t^4}}, \{t, 0, z\}\right]}{\text{EllipticK}\left[\frac{1}{2}\right]}$$
]
```

Unaided, *PolarMap* does not deal with this example very effectively; it cuts off the corners of the square. We can improve matters by substituting a correct *Line* for the incorrect one.

```
In[52]:= distortion = ReplacePart[PolarMap[f, {0, 1}, {0, 2 π}, Lines → {15, 25},  
    PlotRange → All, Axes → False, DisplayFunction → Identity],  
    Line[{{1, -1}, {1, 1}, {-1, 1}, {-1, -1}, {1, -1}}, {1, 15, 1, 1}];
```

```
In[53]:= DisplayTogetherArray[{originaldisk, distortion}]
```

From In[53]:=



For this example, $\rho(\phi)$ is a function of period $\pi/2$ that is equal to $\sec(\phi)$ for $-\pi/4 \leq \phi \leq \pi/4$, and can therefore be defined by

```
In[54]:= ρ[ϕ_] := Sec[Mod[ϕ +  $\frac{\pi}{4}$ ,  $\frac{\pi}{2}$ ] -  $\frac{\pi}{4}$ ]
```

As before, Theodorsen's integral equation is solved for the function $\phi(\theta)$, and the exact solution for $-\pi/4 \leq \phi \leq \pi/4$ is now

```
In[55]:= exactsoln[θ_] := ArcTan[
$$\frac{\text{EllipticF}[\text{ArcSin}[\sqrt{2} \text{Sin}[\theta]], \frac{1}{2}]}{\text{EllipticK}[\frac{1}{2}]}$$
]
```

Again we begin with

```
In[56]:= approxsoln[θ_] = θ;
```

To give a visual assessment of the accuracy we now use

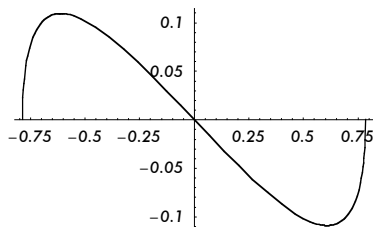
```
In[57]:= errorplot :=
```

```
Plot[exactsoln[θ] - approxsoln[θ], {θ, -π/4, π/4}, PlotRange -> All]
```

The initial approximation is not very close.

```
In[58]:= errorplot
```

From In[58]:=



Because of the symmetry, we can concentrate on the interval $-\pi/4 < \theta < \pi/4$; we know that $\phi(\theta)$ is equal to θ plus an odd function of θ that must vanish at both $\theta = -\pi/4$ and $\theta = \pi/4$. Since these points correspond to the corners of the square, it is near them that difficulties arise, but a local analysis shows that $\phi(\theta)$ has square root singularities and an unbounded derivative at these points. Thus, in spite of the periodicity, trigonometric interpolation is not appropriate for this example and we use `Fit` with a more apt set of basis functions for our representation. A suitable set is given by

```
In[59]:= basis =
```

$$\text{Union}\left[\text{Table}\left[\left(\frac{\pi^2}{16} - \theta^2\right) \theta^{2i-1}, \{i, 4\}\right], \text{Table}\left[\theta \left(\frac{\pi^2}{16} - \theta^2\right)^{\frac{1}{2}(2i-1)}, \{i, 4\}\right]\right]$$

$$\text{Out[59]} = \left\{ \theta \sqrt{\frac{\pi^2}{16} - \theta^2}, \theta \left(\frac{\pi^2}{16} - \theta^2\right), \theta^3 \left(\frac{\pi^2}{16} - \theta^2\right), \theta^5 \left(\frac{\pi^2}{16} - \theta^2\right), \right.$$

$$\left. \theta^7 \left(\frac{\pi^2}{16} - \theta^2\right), \theta \left(\frac{\pi^2}{16} - \theta^2\right)^{3/2}, \theta \left(\frac{\pi^2}{16} - \theta^2\right)^{5/2}, \theta \left(\frac{\pi^2}{16} - \theta^2\right)^{7/2} \right\}$$

To try to obtain a uniform distribution of errors we should, as before, concentrate the interpolation points near the ends of the interval, and therefore introduce a nonlinear scaling similar to one used before.

```
In[60]:= new[θ_] := 1/4 π Sin[2 θ]
```

In fact, because of the extra difficulties we now have at the ends, we will push points even closer to them by using `new[new[θ]]`. (If larger errors near the ends are not a problem, you can obtain smaller errors over the major part of the interval by just using `new[θ]`.) We use `iterstep` constructed much as before; it produces from the interpolation on $-\pi/4 < \theta < \pi/4$ an `approxsoln[θ]` that is applicable for all values of θ .

```

In[61]:= iterstep := (values = Table[
    {new[new[θ]], 1/(2 π) CauchyPrincipalValue[Log[ρ[approxsoln[τ]]]
    Cot[1/2 (new[new[θ]] - τ)], {τ, 0, {new[new[θ]]}, π/4,
    3 π/4, 5 π/4, 7 π/4, 2 π}}], {θ, π/36, π/4 - π/36, π/36}];
partapproxsoln[θ_] = Fit[values, basis, θ];
approxsoln[θ_] = θ + partapproxsoln[Mod[θ + π/4, π/2] - π/4]

```

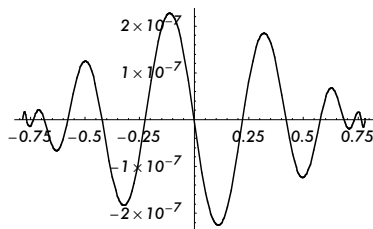
Satisfactory results are now obtained with

```

In[62]:= Do[iterstep, {50}]; errorplot

```

From In[62]:=



Our approximation for the mapping is given by

```

In[63]:= approxf[z_] :=
  Chop[If[Abs[z] == 1, ρ[approxsoln[Arg[z]]] e^i approxsoln[Arg[z]],
  z Exp[1/(2 π) NIntegrate[(Log[ρ[approxsoln[θ]]] +
    i (approxsoln[θ] - θ))/(1 - z e^-i θ), Evaluate[Flatten[
    {θ, Sort[N[{-π, -3 π/4, -π/4, Arg[z], π/4, 3 π/4, π}]]}]]]]]]

```

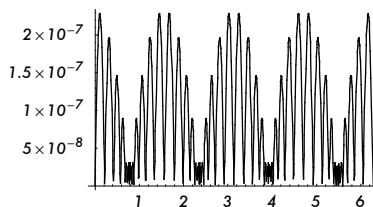
This is much as before, but *Mathematica* now appreciates being warned to expect trouble with `NIntegrate` at points corresponding to the corners. We can test the error at various points as in the previous example. This gives a plot of the absolute value of the error on the boundary.

```

In[64]:= Plot[Abs[error[e^i θ]], {θ, 0, 2 π}]

```

From In[64]:=



And here we obtain the maximum absolute value of the error at a random distribution of interior points.

```
In[65]:= Table[error[Random[] ei Random[Real, {0, 2 π}]], {100}] // Abs // Max
```

```
Out[65]:= 8.19077 × 10-8
```

■ Nekrasov's Equation

An important integral equation in hydrodynamics is that derived by Nekrasov [12] to describe waves moving on the surface of a fluid. Assuming the fluid to be inviscid and the flow to be irrotational, the form of this equation that is relevant when the depth of the fluid is infinite is

$$\phi(s) = \frac{1}{3\pi} \int_{-\pi}^{\pi} \frac{\mu \sin(\phi(t))}{1 + \mu \int_0^t \sin(\phi(u)) du} \log\left(\frac{1}{|2 \sin(\frac{s-t}{2})|}\right) dt.$$

Here the dependent variable ϕ is the angle between the tangent to the wave surface and the horizontal, while the independent variable s is the velocity potential of the flow along the wave surface. We assume that we are dealing with a periodic wave train and translating with the wave so that it has a fixed shape in our frame of reference, and $\phi(s)$ then has 2π as a period. With $s = 0$ corresponding to the crest of a wave, so $\phi(0) = 0$, it can be shown that $\phi(s)$ is necessarily an odd function of s if there is just one crest per period, and the waves are therefore symmetric about their crests with $s = -\pi$ and $s = \pi$ corresponding to troughs; thus $\phi(-\pi) = \phi(\pi) = 0$. Granted this, it is usual to deal with the above equation in the slightly modified form

$$\phi(s) = \frac{1}{3\pi} \int_0^{\pi} \frac{\mu \sin(\phi(t))}{1 + \mu \int_0^t \sin(\phi(u)) du} \log\left(\left|\frac{\sin(\frac{s+t}{2})}{\sin(\frac{s-t}{2})}\right|\right) dt.$$

The parameter μ is given in terms of physical variables by

$$\mu = \frac{3 g \lambda c}{2 \pi Q^3},$$

where g is the acceleration due to gravity, λ is the wavelength, c is the speed with which the wave is progressing (since we are moving with the wave, this is actually the speed of the fluid at infinite depth), and Q is the fluid speed at the crest.

Whatever the value of μ , there is always the solution $\phi(s) = 0$, and this is the only solution for small values μ . As μ increases, there is a bifurcation at $\mu = 3$ with a genuine waveform having just one crest per period appearing. Because of the nature of s , increasing s from $-\pi$ to π corresponds to moving from right to left along the wave surface, with the usual orientation of axes, so this solution has $\phi(s) > 0$ for $0 < s < \pi$ and $\phi(s) < 0$ for $-\pi < s < 0$. For larger μ , there are other possible solutions corresponding to there being several crests between $s = -\pi$ and $s = \pi$, but we can select the one with just one crest there by choosing an initial approximation with the correct variation. We use

```
In[66]:= approxsoln[s_] = 0.1 Sin[s];
```

There is a logarithmic singularity of the integrand in this form of Nekrasov's equation. This can be transformed into a removable singularity by performing an integration by parts but, as with Theodorsen's equation, *Mathematica* seems to prefer the singular form provided it is given a little help.

With this example, we will compromise on accuracy to avoid excessive computation times; we shall see that the problem becomes more difficult as μ increases, and any method of solution then requires lengthy procedures to obtain accurate results. With this in mind, we use `NDSolve` to compute the integral in the denominator of the integrand as a function of t just once during each iteration, rather than using `NIntegrate` every time we evaluate this integrand. The basic philosophy behind the iteration is much as before; we use

```

In[67]:= k[s_, t_] := Log[Abs[ $\frac{\text{Sin}[\frac{s+t}{2}]}{\text{Sin}[\frac{s-t}{2}]}$ ]]

In[68]:= iterstep := (Clear[denomint]; denomint[t_] =
    denomint[t] /. NDSolve[{denomint'[t] == Sin[approxsoln[t]],
        denomint[0] == 0}, denomint[t], {t, 0,  $\pi$ }]][[1];
    values = Table[{new[s],  $\frac{1}{3\pi}$  NIntegrate[
         $\frac{k[\text{new}[s], t] \mu \text{Sin}[\text{approxsoln}[t]]}{1 + \mu \text{denomint}[t]}$ ,
        {t, 0, new[s],  $\pi$ }, SingularityDepth  $\rightarrow$  40,
        MaxRecursion  $\rightarrow$  40}], {s,  $\frac{\pi}{n}$ ,  $\pi - \frac{\pi}{n}$ ,  $\frac{\pi}{n}$ ]];
    approxsoln = Interpolation[Union[values,
        (values /. {x_?NumericQ, y_?NumericQ}  $\rightarrow$  {-x, -y}),
        {{- $\pi$ , 0}, {0, 0}, { $\pi$ , 0}}]])
```

We have `Interpolation` here rather than `InterpolatingPolynomial` or `Fit` for reasons to be given later. We also have an integer n governing the number of interpolation points we use, and `new[s]` to allow us to use a scaling to redistribute these points, but we begin with a moderate value of the former and no scaling for the latter.

```

In[69]:= n = 10; new[s_] := s
```

Now for a moderate value of μ .

```

In[70]:=  $\mu = 5$ ;
```

We obtain the relevant approximation and save it for later use.

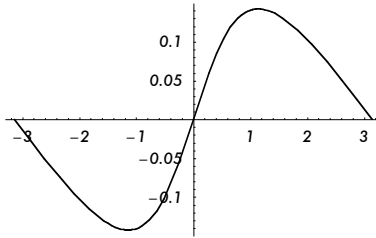
```

In[71]:= Do[iterstep, {30}]; soln[1, s_] = approxsoln[s];
```

We can now plot the result.

```
In[72]:= Plot[soln[1, s], {s, -π, π}]
```

From In[72]:=

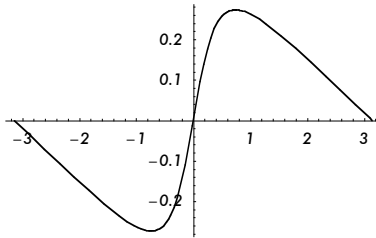


We see that there has been only moderate distortion of our original approximation as part of a sine wave, but there is a steepening near the origin. Anticipating that there will be more such distortion for larger μ , we increase the value of n for the next case.

```
In[73]:= μ = 10; n = 20; Do[iterstep, {30}];
```

```
In[74]:= soln[2, s_] = approxsoln[s]; Plot[soln[2, s], {s, -π, π}]
```

From In[74]:=



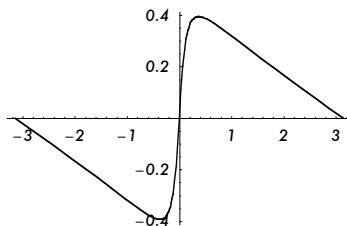
The distortion has indeed increased somewhat and we note that the solution looks roughly like a cubic near $s = 0$, but is virtually linear towards the ends of the interval. Neither a polynomial nor a finite trigonometric sum is suitable for approximating such a function, which is why we have used the cubic splines produced by `Interpolation`. Since it is now clear that the neighborhood of the origin is becoming more important, we will introduce a scaling that pushes more interpolation points into this region as we increase μ further, and also increase n .

```
In[75]:= μ = 30; n = 30; new[s_] := π (s/π)^1.4;
```

```
Do[iterstep, {30}]; soln[3, s_] = approxsoln[s];
```

```
In[76]:= Plot[soln[3, s], {s, -π, π}, PlotRange -> All]
```

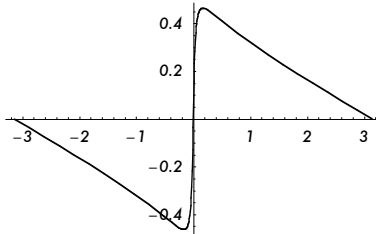
From In[76]:=



We will perform the calculations for just one more value of μ , modifying both n and `new[s]` again.

```
In[77]:=  $\mu = 100$ ;  $n = 50$ ; new[s_] :=  $\pi \left(\frac{s}{\pi}\right)^{1.5}$ ;
          Do[iterstep, {30}]; soln[4, s_] = approxsoln[s];
In[78]:= Plot[soln[4, s], {s, - $\pi$ ,  $\pi$ }, PlotRange -> All]
```

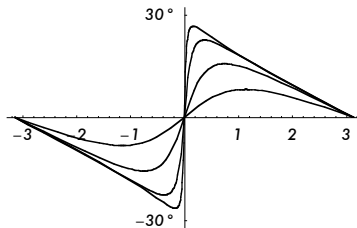
From In[78]:=



We can now bring together the four solutions computed thus far.

```
In[79]:= Plot[Evaluate[Table[soln[i, s], {i, 4}]],
          {s, - $\pi$ ,  $\pi$ }, PlotRange -> {-0.56, 0.56},
          Ticks -> {Automatic, {{ $\frac{\pi}{6}$ , "30°"}, {- $\frac{\pi}{6}$ , "-30°"}}}]
```

From In[79]:=



The trend seems to be clear; as μ increases further, we might expect the solution to approach one where this graph has a discontinuity at $s = 0$. This would mean that the surface of the wave is no longer smooth at the crest, but has a corner there. Now Stokes showed that one could have such corners only if the angle (as seen by the fluid) is 120° , which here corresponds to ϕ jumping from -30° to $+30^\circ$ as s increases through 0, and conjectured that the limit of this family of waveforms as $\mu \rightarrow \infty$ is, indeed, such a wave with a 120° corner. This is supported by the fact that the flow in such a corner has zero fluid speed actually at the corner, for letting $Q \rightarrow 0$ in the definition of μ above we obtain $\mu \rightarrow \infty$. This Stokes's conjecture (now proved) is also illustrated by the previous graph.

We can use the solutions obtained to plot the actual shape of the waves. For convenience, we normalize the wavelengths to 2 and choose the horizontal axis to pass through the troughs, so the profiles covering one period start at the point $(-1, 0)$ and end at the point $(1, 0)$.

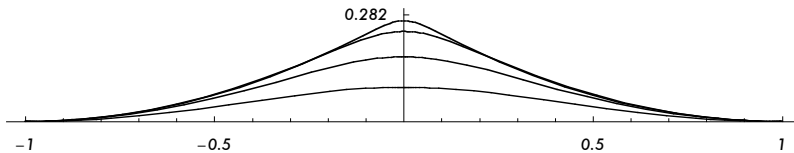
```

In[80]:= Do[ $\mu = 60 - \frac{280 i}{3} + \frac{85 i^2}{2} - \frac{25 i^3}{6}$ ;
Clear[denomint, dtau, tau, y];
denomint[s_] =
denomint[s] /. NDSolve[{denomint'[s] == Sin[soln[i, s]],
denomint[0] == 0}, denomint[s], {s, - $\pi$ ,  $\pi$ }]][1];
dtau[s_] =  $\frac{\mu \text{Sin}[\text{soln}[i, s]]}{3 (1 + \mu \text{denomint}[s])}$ ;
tau[s_] = tau[s] /.
NDSolve[{tau'[s] == dtau[s], tau[0] == 0}, tau[s], {s, - $\pi$ ,  $\pi$ }]][1];
y[s_] = y[s] /. NDSolve[{y'[s] ==  $e^{i \text{soln}[i, s] - \text{tau}[s]}$ , y[0] == 0},
y[s], {s, - $\pi$ ,  $\pi$ }]][1];
zed[s_] =  $\frac{y[s] - i \text{Im}[y[-\pi]]}{\text{Re}[y[-\pi]]}$ ;
waveplot[i] = ParametricPlot[{Re[zed[s]], Im[zed[s]]},
{s, - $\pi$ ,  $\pi$ }, DisplayFunction -> Identity], {i, 4}]

In[81]:= Show[Evaluate[Table[waveplot[i], {i, 4}]],
AspectRatio -> Automatic, Ticks -> {Automatic, {0.282}},
PlotRange -> {0, 0.3}, DisplayFunction -> $DisplayFunction]

```

From In[81]:=



Again we see evidence that the waveform is approaching a state with a corner as $\mu \rightarrow \infty$. The point marked 0.282 on the vertical axis is where the corner forming the crest lies in this limiting wave of greatest height. (To illustrate the difficulty of calculations in this area, however they are done and whatever theoretical approach is used, while there is agreement on this value to 3 decimal places in the literature, there still seems to be some doubt over what the fourth digit should be.)

The previous two graphs also suggest a further conjecture, due to Krasovskii, to the effect that, for all waves of this family, we have $|\phi(s)| \leq 30^\circ$ for all values of s . In fact, this conjecture is now known to be false. For large values of μ , the graph of $\phi(s)$ acquires small ripples near the maximum and minimum in our figure, and these are responsible for values of $|\phi(s)|$ that are slightly greater than 30° at points close to the crest. To demonstrate these in a numerical approach based on Nekrasov's equation, calculations performed with $\mu = 10^{18}$ showed a maximum value for $|\phi(s)|$ of 30.3787° . Since the effect being investigated looks very like the familiar Gibbs phenomenon associated with approximations, one must be careful to produce a real effect and not one artificially introduced by the method used. With such large values of μ involved, it is evident that asymptotic methods are more appropriate for such investigations. A recent article concerning this problem is Byatt-Smith [13], and earlier references can be traced from this source.

■ Concluding Remarks

In our examples, we have been able to use the natural iteration scheme, but it is useful to know that a slight modification can yield convergence when this fails. With a linear integral equation, whose general form we write as

$$f(x) = g(x) + \lambda \int_a^b K(x, y) f(y) dy,$$

the standard iteration leading to the Neumann series is guaranteed to converge for any initial guess only if $|\lambda|$ is less than the modulus of the smallest eigenvalue. But Bückner [14] has shown that if we use instead the iteration

$$f_{n+1}(x) = (1 - \theta) g(x) + \theta f_n(x) + (1 - \theta) \lambda \int_a^b K(x, y) f_n(y) dy,$$

with θ some suitable constant, we can achieve convergence for λ over a greater range of values; this reduces to the standard iteration if we set $\theta = 0$. As an example of this, consider the equation

$$f(x) = e^x + \lambda \int_0^1 e^{x-y} f(y) dy,$$

which has the single eigenvalue $\lambda = 1$, and the solution for all $\lambda \neq 1$ is

$$\text{In}[82]:= \text{exactsoln}[x_]:= \frac{e^x}{1-\lambda}$$

We begin with

```
In[83]:= Clear[approxsoln]; approxsoln[x_] = 1;
```

and use the same means of testing accuracy as previously with

```
In[84]:= errorplot :=
  Plot[exactsoln[x] - approxsoln[x], {x, 0, 1}, PlotRange -> All]
```

The procedure is much as used earlier, with

```
In[85]:= new[x_] := Sin[ $\frac{\pi x}{2}$ ]2
In[86]:= iterstep :=
  (values = Table[{new[x], (1 -  $\theta$ ) enew[x] +  $\theta$  approxsoln[new[x]] + (1 -  $\theta$ )  $\lambda$ 
    NIntegrate[enew[x]-y approxsoln[y], {y, 0, 1}], {x, 0, 1,  $\frac{1}{10}$ }}];
  approxsoln[x_] = InterpolatingPolynomial[values, x])
```

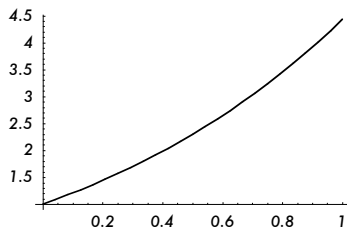
For the value $\lambda = 0.5$, the usual iteration with $\theta = 0$ converges.

```
In[87]:=  $\lambda = 0.5$ ;  $\theta = 0$ ;
```

The initial approximation is not very good.

```
In[88]:= errorplot
```

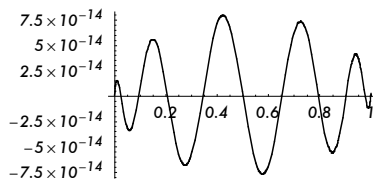
From In[88]:=



But we easily improve on it.

```
In[89]:= Do[iterstep, {50}]; errorplot
```

From In[89]:=



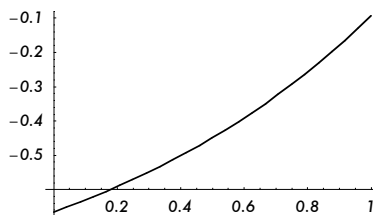
Now try the same routine with $\lambda = -2$.

```
In[90]:= approxsoln[x_] = 1;  $\lambda = -2$ ;  $\theta = 0$ ;
```

First a look at the initial error.

```
In[91]:= errorplot
```

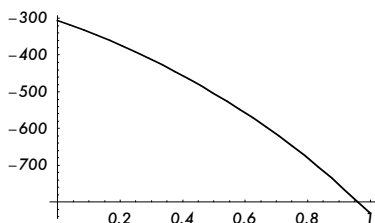
From In[91]:=



Now matters are *not* improved by this iteration!

```
In[92]:= Do[iterstep, {10}]; errorplot
```

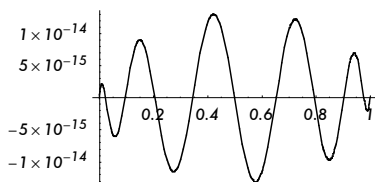
From In[92]:=



But if we use $\theta = 0.6$ in the scheme instead of $\theta = 0$, convergence is restored.

```
In[93]:= approxsoln[x_] = 1;  $\lambda = -2$ ;  $\theta = 0.6$ ; Do[iterstep, {70}]; errorplot
```

From In[93]:=



■ References

- [1] R. Kress, *Linear Integral Equations*, New York: Springer-Verlag, 1989.
- [2] R. Kress, *Numerical Analysis*, New York: Springer-Verlag, 1998.
- [3] L. Fox and E. T. Goodwin, "The Numerical Solution of Non-Singular Linear Integral Equations," *Philosophical Transactions of the Royal Society of London, Series A*, **245**, 1953 pp. 501–534.
- [4] E. R. Love, "The Electrostatic Field of Two Equal Circular Conducting Disks," *Quarterly Journal of Mechanics and Applied Mathematics*, **2**, 1949 pp. 428–451.
- [5] J. C. Cooke, "The Coaxial Circular Disc Problem," *Zeitschrift für Angewandte Mathematik und Mechanik*, **38**, 1958 pp. 349–356.
- [6] E. R. Love, "The Potential Due to a Circular Parallel Plate Condenser," *Mathematika*, **37**, 1990 pp. 217–231.
- [7] W. C. Chew and J. A. Kong, "Microstrip Capacitance for a Circular Disk through Matched Asymptotic Expansions," *SIAM Journal on Applied Mathematics*, **42**, 1982 pp. 302–317.
- [8] V. I. Fabrikant, "Electrostatic Problem of Several Arbitrarily Charged Unequal Coaxial Disks," *Journal of Computational and Applied Mathematics*, **18**, 1987 pp. 129–147.
- [9] P. Henrici, *Applied and Computational Complex Analysis*, Vol. 3, New York: John Wiley & Sons, 1986.
- [10] P. K. Kythe, *Computational Conformal Mapping*, Boston: Birkhäuser, 1998.

- [11] H. Kober, *Dictionary of Conformal Representations*, New York: Dover, 1957.
- [12] A. I. Nekrasov, "On Waves of Permanent Type," *Izv. Ivanovo-Voznesensk. Politekhn. Inst.*, **3**, 1921 pp. 52–65; **6**, 1922 pp. 155–171.
- [13] J. G. Byatt-Smith, "Numerical Solution of Nekrasov's Equation in the Boundary Layer Near the Crest for Waves Near the Maximum Height," *Studies in Applied Mathematics*, **106**, 2001 pp. 393–405.
- [14] H. Bückner, "A Special Method of Successive Approximations for Fredholm Integral Equations," *Duke Mathematical Journal*, **15**, 1948 pp. 197–206.

About the Author

Stan Richardson received his Ph.D. in applied mathematics from the University of Cambridge in 1968 and has been at Edinburgh since 1971. His principal research interest is in free boundary problems, particularly those arising in fluid mechanics, the approach being essentially analytic using methods based on conformal mapping and complex variable theory.

Stan Richardson

*School of Mathematics
University of Edinburgh
James Clerk Maxwell Building
The King's Buildings
Mayfield Road
Edinburgh EH9 3JZ
Scotland
S.Richardson@ed.ac.uk*

Algebraic Programming in Mathematica

Andrzej Kozłowski

“Algebraic Programming” means programming based on the use of algebraic concepts, such as commutativity, associativity, and distributivity. It is a method of programming for which *Mathematica* is particularly well suited. We give several examples of algebraic programming in *Mathematica*, culminating with a solution of a problem that was posted to the *Math-Group* email list.

■ 1. What is Algebraic Programming?

I first came across the expression “the algebraic method in programming” (or algebraic programming for short) in the context of *Mathematica* in the book by Ilan Vardi [1]. By algebraic programming, Vardi meant solving programming problems by using concepts of abstract algebra, such as distributivity, commutativity, and associativity. Vardi gives some examples of algebraic programming. One of these is a solution to the following problem.

Q: Write a program that will produce all possible subsets of a given set (list) of symbols, for example $\{a, b, c\}$.

Vardi makes the observation that the answer to the problem is contained in the following algebraic formula.

```
In[1]:= s = (1 + a) (1 + b) (1 + c);
```

```
In[2]:= Expand[s]
```

```
Out[2]= 1 + a + b + a b + c + a c + b c + a b c
```

This formula depends only on the algebraic property of distributivity of multiplication and the existence of a unit. Indeed, *Mathematica* makes it possible to solve the problem directly by converting the last formula.

```
In[3]:= Variables /@ (Expand[s] /. Plus -> List) // Sort
```

```
Out[3]= {{}, {a}, {b}, {c}, {a, b}, {a, c}, {b, c}, {a, b, c}}
```

We can write a function that will apply this method to any set of symbols. Note, however, that this program will not work if we use a list of numbers!

```
In[4]:= subSets[s_List] :=
      Variables /@ (Expand[Times @@ (1 + s)] /. Plus -> List) // Sort

In[5]:= subSets[{a, b, c, d}]

Out[5]= {{}, {a}, {b}, {c}, {d}, {a, b}, {a, c}, {a, d}, {b, c}, {b, d},
      {c, d}, {a, b, c}, {a, b, d}, {a, c, d}, {b, c, d}, {a, b, c, d}}
```

This attractive method can be used to produce unexpected solutions to surprisingly many problems. However, this is not actually what Vardi meant by “algebraic programming.” In fact one might call this approach “naive algebraic programming.” It is naive because it makes use of specific algebraic functions rather than the general algebraic principles on which they are built. Here is a “sophisticated” algebraic program due to Vardi that achieves the same result.

```
In[6]:= subSetsV[s_List] := Distribute[{{}}, {#}] & /@ s, List, List, Union]

In[7]:= subSetsV[{a, b, c, d}]

Out[7]= {{}, {d}, {c}, {c, d}, {b}, {b, d}, {b, c}, {b, c, d}, {a}, {a, d},
      {a, c}, {a, c, d}, {a, b}, {a, b, d}, {a, b, c}, {a, b, c, d}}
```

Not surprisingly, the second program not only works with symbols and numbers, but is also considerably faster than the first.

```
In[8]:= test = Array[a, {16}];

In[9]:= subSets[test]; // Timing

Out[9]= {9.37 Second, Null}

In[10]:= subSetsV[test]; // Timing

Out[10]= {3.13 Second, Null}
```

The main reason is that the second program is more “abstract,” in the sense that it makes fewer assumptions and has to perform fewer checks. Essentially it applies the principle of “distributivity” in the widest possible sense. In general, algebraic computations using functions like `Times`, `Expand`, and so forth involve extensive sorting of intermediate expressions before they are brought to a canonical form. From the point of view of efficiency, it is much better to operate with lists because no evaluation or canonicalization ever happens, and lists can contain anything.

■ 2. Coefficients of Polynomials and Series

Probably the best known use of algebraic programming is in solving combinatorial problems, where many solutions can be expressed in terms of so-called “generating functions.” Many examples are considered in [2]. Here I shall only briefly mention a problem that was sent to *MathGroup* in July 2001.

Q: Define a *Mathematica* function that counts the number of ways of partitioning a number into a fixed number of parts (repetitions allowed) taken from a given set.

Here is a solution that uses algebraic programming.

```
In[11]:= kpartitions[n_, k_, l_List] :=
SeriesCoefficient[Series[SeriesCoefficient[
Series[Times @@ ((1/(1 - t*x^#)) & /@ l), {x, 0, n}], n],
{t, 0, k}], k]
```

Here is an example.

```
In[12]:= kpartitions[24, 5, {1, 2, 3, 4, 7, 11, 17, 19}]
```

```
Out[12]= 13
```

The proof of the correctness of this solution is left to the reader. The above solution, while elegant, suffers from the usual defects of naive algebraic programming; it is fairly slow. Much faster solutions can be found in the *MathGroup* archives.

■ 3. Attributes in *Mathematica*

Let us look at another example. Suppose we are given a list of n objects and want to construct all permutations of k objects from them. Again, we shall look for an algebraic program to realize this task. Let us consider an example where $n = 4$, $k = 3$, and the objects are $\{a, b, c, d\}$. Then, as before, we see that we would get an answer by taking the terms of the expansion of

$$(a + b + c + d)(a + b + c + d)(a + b + c + d)$$

except that this time we do not want the multiplication to be commutative. *Mathematica* has the following function.

```
In[13]:= ? NonCommutativeMultiply
```

`a ** b ** c` is a general associative,
but non-commutative, form of multiplication. More...

However it knows very few rules so if we are lazy we might just clear the `Orderless` attribute from `Times` and do something like this.

```
In[14]:= Perms[S_List, k_Integer] :=
Module[{perms}, ClearAttributes[Times, Orderless];
perms = Flatten /@ (Expand[(Plus @@ S)^3] /
{Plus -> List, Times -> List, Power[x_, n_] -> Table[x, {n}]});
SetAttributes[Times, Orderless]; perms]
```

```
In[15]:= Perms[{a, b, c, d}, 3]
```

```
Out[15]= {{a, b, a}, {c, b, a}, {d, b, a}, {b, b, a}, {a, c, a},
  {b, c, a}, {d, c, a}, {c, c, a}, {a, d, a}, {b, d, a},
  {c, d, a}, {d, d, a}, {b, a, a}, {c, a, a}, {d, a, a}, {a, a, a},
  {b, a, b}, {c, a, b}, {d, a, b}, {a, a, b}, {a, c, b}, {b, c, b},
  {d, c, b}, {c, c, b}, {a, d, b}, {b, d, b}, {c, d, b}, {d, d, b},
  {a, b, b}, {c, b, b}, {d, b, b}, {b, b, b}, {b, a, c}, {c, a, c},
  {d, a, c}, {a, a, c}, {a, b, c}, {c, b, c}, {d, b, c}, {b, b, c},
  {a, d, c}, {b, d, c}, {c, d, c}, {d, d, c}, {a, c, c}, {b, c, c},
  {d, c, c}, {c, c, c}, {b, a, d}, {c, a, d}, {d, a, d}, {a, a, d},
  {a, b, d}, {c, b, d}, {d, b, d}, {b, b, d}, {a, c, d}, {b, c, d},
  {d, c, d}, {c, c, d}, {a, d, d}, {b, d, d}, {c, d, d}, {d, d, d}}
```

As before, this will not work for numbers instead of symbols. However, we could easily deal with this problem by mapping `Hold` on the list and then applying `ReleaseHold` at the end. Another problem is that we temporarily cleared the `Orderless` attribute of `Times`, which is generally not considered a very good idea and may produce unexpected results. In this case there were no such surprises. However, a general algebraic version of the code is both simpler and much more efficient.

```
In[16]:= Perms1[S_List, k_Integer] := Distribute[Table[S, {k}], List]
```

```
In[17]:= Perms1[{a, b, c, d}, 3]
```

```
Out[17]= {{a, a, a}, {a, a, b}, {a, a, c}, {a, a, d}, {a, b, a},
  {a, b, b}, {a, b, c}, {a, b, d}, {a, c, a}, {a, c, b},
  {a, c, c}, {a, c, d}, {a, d, a}, {a, d, b}, {a, d, c}, {a, d, d},
  {b, a, a}, {b, a, b}, {b, a, c}, {b, a, d}, {b, b, a}, {b, b, b},
  {b, b, c}, {b, b, d}, {b, c, a}, {b, c, b}, {b, c, c}, {b, c, d},
  {b, d, a}, {b, d, b}, {b, d, c}, {b, d, d}, {c, a, a}, {c, a, b},
  {c, a, c}, {c, a, d}, {c, b, a}, {c, b, b}, {c, b, c}, {c, b, d},
  {c, c, a}, {c, c, b}, {c, c, c}, {c, c, d}, {c, d, a}, {c, d, b},
  {c, d, c}, {c, d, d}, {d, a, a}, {d, a, b}, {d, a, c}, {d, a, d},
  {d, b, a}, {d, b, b}, {d, b, c}, {d, b, d}, {d, c, a}, {d, c, b},
  {d, c, c}, {d, c, d}, {d, d, a}, {d, d, b}, {d, d, c}, {d, d, d}}
```

Here we see a characteristic difference between *Mathematica* and specialized programs for dealing with abstract algebraic structures like rings, groups, and so on. Such programs tend to be object oriented and contain structures like groups or rings as some form of data types. *Mathematica* does not have objects or types that correspond to groups or rings, but contains functions and attributes that let one implement distributivity, commutativity, associativity, and so forth in a functional way. As a bonus, we can find uses for them in more general programming that are not strictly concerned with abstract algebraic structures.

As we have seen, distributivity is implemented in *Mathematica* through the function `Distribute`. How about the other fundamental algebraic notions, commutativity and associativity? These are essentially implemented as a so-called “Attributes” of functions `Orderless` and `Flat`. Understanding these attributes (as well as `OneIdentity`, which does not have a simple algebraic motivation) is the foundation of algebraic programming.

```
In[18]:= ? Orderless
```

Orderless is an attribute that can be assigned to a symbol *f* to indicate that the elements *ei* in expressions of the form *f*[*e*₁, *e*₂, ...] should automatically be sorted into canonical order. This property is accounted for in pattern matching. More...

The attribute `Orderless` in a function is usually said to correspond to the mathematical notion of commutativity. This is indeed true, but one has to be careful with the interpretation of “correspond.” Many functions that are “mathematically” commutative do not have the attribute `Orderless`. Such is the case with the logical function `Or`. From the mathematical viewpoint the function is commutative.

```
In[19]:= << Experimental`
```

```
In[20]:= ImpliesQ[Or[A, B], Or[B, A]]
```

```
Out[20]= True
```

However, `Orderless` is not an attribute of `Or`.

```
In[21]:= Attributes[Or]
```

```
Out[21]= {Flat, HoldAll, OneIdentity, Protected}
```

We can see the effect this has on pattern matching.

```
In[22]:= a || b /. _ || a -> c
```

```
Out[22]= a || b
```

```
In[23]:= b || a /. _ || a -> c
```

```
Out[23]= c
```

The reason is not mathematical but has to do with performance and the way logical functions are used in programming, where one actually may wish to exploit the order of evaluation. The arguments of `Or`[*p*, *q*, *r*, *s*, *t*, ...] are evaluated in order, and `True` will be returned the moment one of them is found to be true. This may be a substantial timesaver. Also, consider the following example, which I owe to Daniel Lichtblau.

```
In[24]:= cond1 := And[Length[a] ≥ 2, a[[2]] == 3];
          cond2 := And[a[[2]] == 3, Length[a] ≥ 2];
```

We can see that the following two evaluations will produce different behavior.

```
In[26]:= cond1 /. a -> {x}
          cond2 /. a -> {x}
```

```
Out[26]= False
```

Part::partd : Part specification a[[2]] is longer than depth of object. More ...

```
Out[27]= False
```

In other words: we have to pay attention to order when pattern matching even though `||` is in fact a commutative operation. This can be exploited in programming both to increase efficiency and to avoid generating unnecessary errors.

On the other hand, it is precisely the effect of the “algebraic” attributes `Orderless`, `Flat`, and `OneIdentity` on pattern matching that makes algebraic programming a surprisingly powerful technique.

Let us then consider these remaining algebraic attributes, `Flat` and `OneIdentity`, which together with the function `Distribute` and the `Orderless` attribute, play the central roles in algebraic programming.

The attribute `Flat`, corresponds to the mathematical property of associativity. Of course, as usual there is more to it than just that.

In[28]:= **?Flat**

`Flat` is an attribute that can be assigned to a symbol `f` to indicate that all expressions involving nested functions `f` should be flattened out. This property is accounted for in pattern matching. More...

In order to understand the effect of the algebraic attributes on pattern matching we shall use the *Mathematica* function `ReplaceList`. Indeed, this function, together with `Distribute` and the algebraic attributes, is the most useful *Mathematica* command in algebraic programming.

In[29]:= **?ReplaceList**

`ReplaceList[expr, rules]` attempts to transform the entire expression `expr` by applying a rule or list of rules in all possible ways, and returns a list of the results obtained. `ReplaceList[expr, rules, n]` gives a list of at most `n` results. More...

Let us try using `ReplaceList` to understand the effect of the `Flat` attribute on pattern matching.

We start by taking a function `f` with no attributes and consider the following replacement rule.

In[30]:= **ClearAll[f]**

In[31]:= **ReplaceList[f[a, b, c], f[a_, b_] → g[a, b]]**

Out[31]= {}

Not surprisingly there was no match. Now give `f` the attribute `Flat`.

In[32]:= **SetAttributes[f, Flat]**

Let us repeat the previous attempt.

In[33]:= **ReplaceList[f[a, b, c], f[a_, b_] → g[a, b]]**

Out[33]= {g[f[a], f[b, c]], g[f[a, b], f[c]]}

Note what happened. The function $f[a, b, c]$ was now rewritten using “associativity” in two ways, as $f[f[a], f[b, c]]$ and $f[f[a, b], f[c]]$. The rule was

then applied but only at the top level. Now suppose we give f the attribute `OneIdentity` in addition to `Flat`.

```
In[34]:= SetAttributes[f, OneIdentity]
```

```
In[35]:= ReplaceList[f[a, b, c], f[a_, b_] → g[a, b]]
```

```
Out[35]= {g[a, f[b, c]], g[f[a, b], c]}
```

You see that all expressions of the type $f[x]$ were replaced just by x . Of course this happens only when pattern matching. If you enter

```
In[36]:= f[x]
```

```
Out[36]= f[x]
```

you still get just $f[x]$, not x .

Indeed:

```
In[37]:= ?OneIdentity
```

`OneIdentity` is an attribute that can be assigned to a symbol f to indicate that $f[x]$, $f[f[x]]$, etc. are all equivalent to x for the purpose of pattern matching. More...

The useful function `ReplaceAllList` works like `ReplaceList` but also works on all subexpressions.

```
In[38]:= ReplaceAllList[expr_, rules_] :=
Module[{i}, Join[ReplaceList[expr, rules],
If[AtomQ[expr], {}, Join@@Table[ReplacePart[expr, #, i] & /@
ReplaceAllList[expr[[i]], rules], {i, Length[expr]}]]]
```

Let us now look at another example of algebraic programming. Of course, the most natural use of algebraic programming is solving algebra problems. So let us first consider an associativity problem.

Q: Suppose g is a nonassociative noncommutative product of any number of elements. Find all the distinct ways of multiplying four elements using g that would be equal if g were associative.

We use a function f , to which we assign the attributes `Flat` and `OneIdentity`.

```
In[39]:= ClearAll[f]; SetAttributes[f, {Flat, OneIdentity}]
```

```
In[40]:= FixedPoint[Union[Flatten[ReplaceAllList[#, f[a_, b_] → g[a, b]]] &,
f[a, b, c, d]]
```

```
Out[40]= {g[a, f[b, c, d]], g[a, g[b, f[c, d]]],
g[a, g[b, g[c, d]]], g[a, g[f[b, c], d]], g[a, g[g[b, c], d]],
g[f[a, b], f[c, d]], g[f[a, b], g[c, d]], g[f[a, b, c], d],
g[g[a, b], f[c, d]], g[g[a, b], g[c, d]], g[g[a, f[b, c]], d],
g[g[a, g[b, c]], d], g[g[f[a, b], c], d], g[g[g[a, b], c], d]}
```

`In[41]:= Union[% /. f -> g]`

`Out[41]= {g[a, g[b, g[c, d]]], g[a, g[g[b, c], d]],
g[a, g[b, c, d]], g[g[a, b], g[c, d]],
g[g[a, g[b, c]], d], g[g[g[a, b], c], d], g[g[a, b, c], d]}`

This sort of thing is in fact quite useful in various areas of mathematics. Later we shall give just one application of exactly this idea to solving a rather well-known puzzle that is frequently asked on symbolic algebra email lists.

■ 4. The 4 Fours Problem

The following problem was sent to the *MathGroup* email list in June 2002.

Q: Find the numbers 0 to 99 using any rational operator (i.e., an operation which results in a rational number) and 4 fours, where each equation must contain no more and no less than 4 fours. Not all numbers may be possible. Some examples include:

$$0 = 4 + 4 - 4 - 4$$

$$1 = 4/4 + (4 - 4)$$

$$2 = 4/4 + 4/4$$

$$27 = 4! + 4/4 + \sqrt{4}.$$

The problem as formulated is rather unclear because of the vagueness of the phrase “rational operator.” The question allowed not only binary operations but also unary operations, such as taking the square root or factorials. This of course means that there is an infinite number of possible solutions unless one imposes additional restrictions. However, we shall first simplify the problem and consider only binary operations: addition, subtraction, multiplication, division, and exponentiation. Let us try to solve the following simpler version of the problem.

Q: Find all integers between 1 and 99 that are expressible using 4 fours and only the arithmetical operations: +, -, *, /, and ^.

This is very easy to implement in *Mathematica* using algebraic programming. In fact, we shall use exactly the same method as in the last example.

The idea is very simple. First we find all ways of writing expressions like $A_0[A_1[4, 4], A_2[4, 4]]$, where the A_i are the binary operations Times, Plus, Divide, Subtract, and Power. Actually in *Mathematica*, Times and Plus are n -ary operations for any $n \geq 1$, but Divide, Subtract, and Power are not. To handle them all simultaneously, we consider them to be binary operations.

It is clear that the number of different expressions that can be made up of four arguments and binary operations $A_0, A_1, \dots$ (which may be different or the same)

can be described as follows. Let A be an associative binary operation. Then expressions such as $A[x, A[y, A[z, w]]]$ and $A[A[x, y], A[z, w]]$ are equal. Clearly the number of such expressions that are equal by virtue of the associativity of A (which we found in the previous section) is equal to the number of distinct ways of applying three binary functions to four arguments. In fact we can obtain all the expressions of the latter type from those of the former as follows.

First we repeat what we did above, that is, define an associative operation f .

```
ClearAll[f];
SetAttributes[f, {Flat, OneIdentity}]
```

Next, just as above, we find all the ways of “associating” four elements.

```
In[43]:= ls = Union[
  FixedPoint[Union[Flatten[ReplaceAllList[#, f[a_, b_] → g[a, b]]] &,
    f[a, b, c, d]] /. f → g]
```

```
Out[43]= {g[a, g[b, g[c, d]]], g[a, g[g[b, c], d]],
  g[a, g[b, c, d]], g[g[a, b], g[c, d]],
  g[g[a, g[b, c]], d], g[g[g[a, b], c], d], g[g[a, b, c], d]}
```

Then we remove all duplicates and expressions that involve the application of g to three elements.

```
In[44]:= ls = DeleteCases[ls, _?(Not[FreeQ[#, g[x_ /; Length[{x}] ≥ 3]]] &), 1]
Out[44]= {g[a, g[b, g[c, d]]], g[a, g[g[b, c], d]],
  g[g[a, b], g[c, d]], g[g[a, g[b, c]], d], g[g[g[a, b], c], d]}
```

Finally we replace g by A_2, A_1, A_0 , in this order, and make all the elements $\{a, b, c, d\}$ equal to 4.

```
In[45]:= ls =
  (i = 0; MapAll[If[# === g, # /. g → Subscript[A, Mod[++i, 3]], #] &, ls,
    Heads → True] /. Thread[Rule[{a, b, c, d}, 4]])
Out[45]= {A1[4, A2[4, A0[4, 4]]], A1[4, A2[A0[4, 4], 4]],
  A1[A2[4, 4], A0[4, 4]], A1[A2[4, A0[4, 4]], 4], A1[A2[A0[4, 4], 4], 4]}
```

(A different approach to the same problem can be found in exercise 13 of Chapter 6 of [2].)

We shall now simply substitute the five binary operations **Plus**, **Subtract**, **Times**, **Divide**, and **Power** for A_0, A_1, A_2 in all possible ways. First we need to redefine **Divide** and **Power**. Basically we do this to avoid error messages caused by division by 0. For later use, we also place a restriction on the maximum and minimum size of the exponent to avoid unnecessary computations of very large and very small numbers.

```
In[46]:= Off[General::"spell1"]; Off[General::"spell"];
power[x_Integer?(# > 0 &), y_Integer?(-16 ≤ # ≤ 16 &)] := x^y;
power[x_] := Indeterminate;
divide[x_, y_] /; y ≠ 0 := Divide[x, y]; divide[x_, 0] := Indeterminate;
```

Next we create a set of rules that we shall use to create all possible ways of replacing the A_i with the binary operations.

```
In[49]:= rules =
  Map[Thread, Map[RuleDelayed[Array[A_#-1 &, {5}], #] &, Distribute[
    Array[{Times, Plus, Subtract, divide, power} &, {5}], List]]];
```

Here is what the list we created looks like.

```
In[50]:= Take[rules, 5]

Out[50]= {{A0 -> Times, A1 -> Times, A2 -> Times, A3 -> Times, A4 -> Times},
  {A0 -> Times, A1 -> Times, A2 -> Times, A3 -> Times, A4 -> Plus},
  {A0 -> Times, A1 -> Times, A2 -> Times, A3 -> Times, A4 -> Subtract},
  {A0 -> Times, A1 -> Times, A2 -> Times, A3 -> Times, A4 -> divide},
  {A0 -> Times, A1 -> Times, A2 -> Times, A3 -> Times, A4 -> power}}
```

In fact, since an actual formula never involves more than three binary operations, we remove the rules involving unnecessary A s.

```
In[51]:= rules = Union[DeleteCases[rules,
  HoldPattern[A3 -> _] | HoldPattern[A4 -> _], Infinity]];
```

```
In[52]:= Take[rules, 5]
```

```
Out[52]= {{A0 -> divide, A1 -> divide, A2 -> divide},
  {A0 -> divide, A1 -> divide, A2 -> Plus},
  {A0 -> divide, A1 -> divide, A2 -> power},
  {A0 -> divide, A1 -> divide, A2 -> Subtract},
  {A0 -> divide, A1 -> divide, A2 -> Times}}
```

Let us now find all possible solutions of the “easy” 4 fours problem.

```
In[53]:= solutions =
  Union[Flatten[Union[Union[Select[Flatten[Union[# /. rules]],
    Element[#, Integers] && 0 ≤ # ≤ 99 &]] & /@ 1s]]]
```

```
Out[53]= {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 12, 15, 16, 17,
  20, 24, 28, 32, 36, 48, 60, 63, 64, 65, 68, 80, 81}
```

```
In[54]:= Length[solutions]
```

```
Out[54]= 27
```

Finally let us find all possible ways of obtaining a given number, for example, 7.

```
In[55]:= patterns =
  Union[Flatten[Union[Flatten[Union[HoldForm[#] /. rules]]] & /@ 1s]]];
```

```
In[56]:= Select[patterns, ReleaseHold[#] == 7 &] /.
  {divide -> Divide, power -> Power}
```

```
Out[56]= {{(4 -  $\frac{4}{4}$ ) + 4, 4 + (4 -  $\frac{4}{4}$ ), 4 - ( $\frac{4}{4}$  - 4), (4 + 4) -  $\frac{4}{4}}$ }
```

This completely solves the “easy” 4 fours problem. We can solve the analogous 5 fives problem and so on in exactly the same way. In the final section we take a look at the “hard” 4 fours problem presented in the original question.

■ 5. The Hard 4 Fours Problem

The hard version of the problem is to find all possible representations of all numbers between 1 and 99 using any “reasonable” operations, including unary ones (“functions”) like `Sqrt` or `Factorial`. Of course there is now an infinite number of possible ways of trying to write out a solution, even with a finite number of operations. We shall therefore concentrate only on representations using `Sqrt` and `Factorial` in addition to the usual arithmetic operations, and only try to find some additional solutions. (If the additional operations were binary [e.g., `Max` or `Min`] then we could use exactly the same method as before to obtain a complete solution.) We first introduce modified definitions of the additional operations.

```
In[57]:= sqrt[x_? (# > 0 &)] := Sqrt[x]; sqrt[x_] := Indeterminate;
        factorial[n_Integer? (0 <= # < 12 &)] = Factorial[n];
        factorial[n_] := Indeterminate;
```

We create a new set of rules that transform our unary functions B_i into `factorial`, `sqrt`, or `Identity` to account for the cases where no unary operation is used.

```
In[59]:= rules1 = Map[Thread, Map[RuleDelayed[Array[B_ &, {4}], #] &, Distribute[
        Array[{Identity, Identity, factorial, sqrt} &, {4}], List]]];
```

```
In[60]:= Take[rules1, 5]
```

```
Out[60]= {{B1 -> Identity, B2 -> Identity, B3 -> Identity, B4 -> Identity},
        {B1 -> Identity, B2 -> Identity, B3 -> Identity, B4 -> Identity},
        {B1 -> Identity, B2 -> Identity, B3 -> Identity, B4 -> factorial},
        {B1 -> Identity, B2 -> Identity, B3 -> Identity, B4 -> sqrt},
        {B1 -> Identity, B2 -> Identity, B3 -> Identity, B4 -> Identity}}
```

We see that there are rather a lot of identical rules. We can reduce their number.

```
In[61]:= rules1 = Union[rules1, SameTest ->
        (((Array[B_ &, {4}] /. #1) == (Array[B_ &, {4}] /. #2)) &)];
```

```
In[62]:= Take[rules1, 5]
```

```
Out[62]= {{B1 -> factorial, B2 -> factorial, B3 -> factorial, B4 -> factorial},
        {B1 -> factorial, B2 -> factorial, B3 -> factorial, B4 -> Identity},
        {B1 -> factorial, B2 -> factorial, B3 -> factorial, B4 -> sqrt},
        {B1 -> factorial, B2 -> factorial, B3 -> Identity, B4 -> factorial},
        {B1 -> factorial, B2 -> factorial, B3 -> Identity, B4 -> Identity}}
```

The most obvious expressions to evaluate are obtained from the earlier ones by replacing all fours by $B_i[4]$.

```
In[63]:= expr[j_] :=
        Module[{i = 0}, MapAll[If[# === 4, Subscript[B, ++i][#], #] &, ls[[j]]]]
```

Here are all the solutions we get in this way.

```
In[64]:= extendedsols = Union[Flatten[
  Union[Select[Flatten[Union[Union[expr[#] /. rules] /. rules1]],
    Element[#, Integers] && 0 ≤ # ≤ 99 &]] & /@ Range[5]]]
Out[64]:= {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,
  21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 34, 35, 36, 37, 38,
  40, 42, 44, 46, 47, 48, 49, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68,
  70, 72, 74, 76, 78, 80, 82, 84, 86, 88, 90, 92, 94, 95, 96, 97, 98}
```

```
In[65]:= Length[%]
```

```
Out[65]= 72
```

An interesting case is the number 31, which was not among the solutions we got earlier. Let us see how it is represented.

```
In[66]:= extendedpatterns = Union[Flatten[
  Union[Union[HoldForm[Evaluate[expr[#]]] /. rules] /. rules1] & /@
  Range[5]]];
```

```
In[67]:= Select[Flatten[extendedpatterns], ReleaseHold[#] == 31 &] /.
  {divide → Divide, power → Power, sqrt → Sqrt,
  factorial → Factorial} /. HoldPattern[Identity[x_]] → x
```

```
Out[67]:= {4! +  $\frac{4+4!}{4}$ , 4! +  $\frac{4!+4}{4}$ ,  $\frac{4+4!}{4} + 4!$ ,  $\frac{4!+4}{4} + 4!$ }
```

Of course there are in principle infinitely many ways to insert more B s into different slots, and it is not clear how many more numbers can be obtained in this way. Here is just one example. Consider the pattern $A_0[A_2[B_2[A_1[B_1[4], 4]], B_3[4]], B_4[4]]$.

```
In[68]:= extrasols = Union[Select[Flatten[Union[
  Union[A_0[A_2[B_2[A_1[B_1[4], 4]], B_3[4]], B_4[4]] /. rules] /. rules1]],
  Element[#, Integers] && 0 ≤ # ≤ 99 &]]
```

```
Out[68]:= {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 18,
  20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 34, 36, 38,
  40, 42, 44, 45, 46, 47, 48, 49, 50, 52, 54, 56, 58, 60, 62, 64,
  66, 68, 70, 72, 74, 76, 78, 80, 82, 84, 88, 90, 92, 94, 96, 98}
```

We see that it produced just one new number.

```
In[69]:= Complement[extrasols, extendedsols]
```

```
Out[69]:= {45}
```

However, we created several new representations for the solutions already found. Here is another representation of 31.

```
In[70]:= extrapatterns = Flatten[Union[
  Union[HoldForm[A_0[A_2[B_2[A_1[B_1[4], 4]], B_3[4]], B_4[4]]] /. rules] /.
  rules1]];
```

```
In[71]:= Select[extrapatterns, ReleaseHold[#] == 31 &] /.
  {divide → Divide, power → Power, sqrt → Sqrt,
  factorial → Factorial} /. HoldPattern[Identity[x_]] → x
```

```
Out[71]:= { $\frac{\frac{4!}{4}! + 4!}{4!}$ ,  $\frac{(\sqrt{4} + 4)! + 4!}{4!}$ ,  $\frac{4! + 4}{4} + 4!$ }
```

Here are all the solutions we have found so far.

```
In[72]:= solutions = Union[extendedsols, extrasols]
```

```
Out[72]= {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,  
21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 34, 35, 36, 37, 38, 40,  
42, 44, 45, 46, 47, 48, 49, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68,  
70, 72, 74, 76, 78, 80, 82, 84, 86, 88, 90, 92, 94, 95, 96, 97, 98}
```

Here are the remaining outstanding numbers.

```
In[73]:= Complement[Range[99], solutions]
```

```
Out[73]= {33, 39, 41, 43, 51, 53, 55, 57, 59, 61, 63, 65, 67,  
69, 71, 73, 75, 77, 79, 81, 83, 85, 87, 89, 91, 93, 99}
```

See if you can find interesting representations of these, possibly using some other operations.

■ Acknowledgments

I would like thank Daniel Lichtblau and Michael Trott [3] for useful comments and suggestions.

■ References

- [1] Ilan Vardi, *Computational Recreations in Mathematica*, Reading, MA: Addison-Wesley, 1991.
- [2] R. L. Graham, D. E. Knuth, and O. Patashnik, *Concrete Mathematics*, Reading, MA: Addison-Wesley, 1989.
- [3] Michael Trott, *The Mathematica Guidebook to Programming*, Berlin: Springer-Verlag, 2003.

About the Author

Andrzej Kozłowski was a professor of mathematics at Toyama International University in Japan when he wrote this article. He is now teaching mathematics and physics at Tokyo Denki University in Japan. Most of his research has been in algebraic topology, but recently he has become interested in mathematical finance, which he is teaching as an Internet-based course at Warsaw University. He recently joined Chatham Research (www.chathamresearch.com) in a project to apply *Mathematica* to problems in quantitative finance and economics. He is a frequent contributor to the *MathGroup* email list.

Andrzej Kozłowski
Tokyo Denki University
Faculty of Information Environment
Chiba, Japan
akoz@mimuw.edu.pl

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
