

The *Mathematica*[®] Journal

Volume 8, Issue 2

■ Contents

□ Departments

Letters	166
Staff and Contributors	168
Calendar of Events	169
<i>Mathematica</i> News	171
New Products	175
Product Profile: web <i>Mathematica</i>	178
New on <i>MathSource</i>	184
New Publications	187

□ Feature Articles

Editor's Introduction by <i>Veikko Keränen</i>	192
The Foundations of Mathematics and <i>Mathematica</i> by <i>Stephen Wolfram</i>	194
Maxillo-Facial Frame: Finite Element Shapes by <i>Gautam Dasgupta and</i> <i>Jacques Treil</i>	235
Theorema: A Proving System Based on <i>Mathematica</i> by <i>Bruno Buchberger</i>	247
Code Generation for Simulation and Control Applications by <i>Mats Fjirstrand and Johan Gunnarsson</i>	253
Musical and Graphic Representations of the Fibonacci String and Proteins Using <i>Mathematica</i> by <i>Erik Jensen and Ronald J. Rusay</i>	278
CFLP: A <i>Mathematica</i> Implementation of a Distributed Constraint Solving System by <i>Mircea Marin, Tetsuo Ida, and Wolfgang Schreiner</i>	287
Behavior and Learning under Law: Automata Containing Evolutionary Algorithms by <i>Seth J. Chandler and Christian J. Jacob</i>	301
Abstracts of Additional Papers from <i>IMS '99</i>	326
Classifieds	348
Article Submissions to <i>The Mathematica Journal</i>	349

The electronic version of this issue is available at www.mathematica-journal.com/issue/v8i2.

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

On the Cover

Mathematica is used to calculate shape functions for convex as well as concave macro-finite elements from the nodal coordinates in the Treil maxillo-facial frame.

Letters

Continued Commitment

Thank you so much for your continued commitment to *Mathematica* subscribers. I am fascinated and amazed by the depth of knowledge and expertise illustrated by those contributing to the *Mathematica* experience. Good job!

Dana Baxter Fadler-Spilios

MathSource® Publication

Do you usually consider notebooks [for publication] that have already been archived by *MathSource*? Thanks.

Qi Zheng

The Editors Reply

No, we usually do not. We consider MathSource to be a prior publication. We encourage authors to submit their articles either to MathSource or TMJ, but not both at once. However, if you still want to submit an article based on packages on MathSource, we will look at it. In your cover letter, please make a note about the MathSource archive numbers.

RealTime 3D Animation

I met one of your *Mathematica* evangelists at the national meeting in New Orleans and he told me that there is a graphics package (which works with Windows at any rate) which gives *Mathematica* click-and-drag functionality for viewing 3D plots. I forgot the name of the package, unfortunately. It was an experimental distribution in *Mathematica* 4.0, I was told, and slated to be standard in *Mathematica* 4.1. Perhaps you would know the name and/or location of this add-on?

Doug Hensley

The Editors Reply

Use <<RealTime3D`. See Roger Germundsson's article in TMJ 7:4 in the section on "User Interface" for an example of how to use it. In Version 4.1 this will "just work" on all platforms. In 4.0 on the Macintosh, you need to install OpenGLLibraries (available from www.apple.com).

Online Subscription Problems

I tried to renew my existing *Mathematica Journal* subscription online by using a link provided me by one of your tech staff. I got the correct form, but it asked for credit card information and yet the page was not handled by a secure server. I can't believe that you don't use secure servers, so I'm reporting this as a problem. Needless to say, I won't renew over a non-secure connection.

Brian Doyle

The Editors Reply

You can be reassured; the subscriptions form uses a secure server. To verify this for yourself, you can go to www.mathematica-journal.com, and then click the link marked "subscriptions." Notice that the broken padlock (or missing padlock for Internet Explorer) turns into a solid gold padlock in the status bar at the bottom of the screen. If you want you can double-click the icon and it will tell you that the page is secure.

A Note to Subscribers

Please send address changes to service@mathematica-journal.com. Your current email address allows us to notify you when the new online issues are available.

Staff and Contributors

Guest Editor: Veikko Keränen (Rovaniemi Polytechnic, Rovaniemi, Finland)

Editor: Flip Phillips, Ph.D. (Skidmore College)

Managing Editor: Joe Grohens

Associate Editor: Greg Springer

Design Team: Jeremy Davis, Linda Kwon, Al Navata

Compositor: Glenn Scholebo

Technical Specialist: Hormozd Gahvari

Archives and Web Production: Jean Buck

Web Development: Ankur Naik, Max Campos

Editorial Staff: George Beck, Catherine Brassac, Marian Peden

Customer Service: Freda Runnells, Cheryl Miller

Columnists: Paul Abbott, Roman Maeder, Michael Trott

Contributors for this Issue: Bruno Buchberger, Seth J. Chandler, Gautam Dasgupta, Johan Gunnarsson, Tetsuo Ida, Christian Jacob, Erik Jensen, Mats Jirstrand, Mircea Marin, Ronald Rusay, Wolfgang Schreiner, Jacques Treil, Stephen Wolfram

Editorial Advisory Board: Steve Christensen (MathTensor, Inc.; Christensen and Associates, Inc.); Donald Glaser (University of California, Berkeley); Donald Knuth (Stanford University); Wolfram Koepf (University Kassel, Germany); Silvio Levy (Mathematical Sciences Research Institute); 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 Michigan); Lynn Steen (St. Olaf College); Francis Sullivan (Institute for Defense Analyses); Bill Thurston (University of California, Davis); Hal Varian (University of California, Berkeley); Rob Wolff (Apple Computer, Inc.); and Stephen Wolfram (Wolfram Research, Inc.).

Customer Services

www.mathematica-journal.com

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

+1-217-398-6000

fax: +1-217-398-1621

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 4* using Janson, Futura, Math CMTT and the standard *Mathematica* fonts.

Publishing

The Mathematica Journal (ISSN 1047-5974) is published quarterly by Wolfram Media, Inc. Annual 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 web site 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.

© 2001 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. *Mathematica* is not associated with Mathematica Policy Research, Inc. or MathTech, Inc. All other product names mentioned are trademarks of their producers. 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 of Events

Professional meetings, conferences, trade shows, exhibits, and *Mathematica* training courses.

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

October

October 11–13, 2001

2001 *Mathematica* Developer Conference

Champaign, Illinois, USA

Info: <http://www.wolfram.com/news/events/devconf2001>

October, 2001

A First Course in *Mathematica*

Wolfram Education *Mathematica* Training

Various cities, USA

October 8–9, Champaign, IL

October 18–19, Baltimore, MD

October 22–23, Dallas, TX

October 22–23, Denver, CO

October 22–24, Denver, CO

October 24, Denver, CO

October 29–30, Phoenix, AZ

Info: <http://www.wolfram.com/services/education/courses/calendar.html>

October 11–12, 2001

Financial Modeling with *Mathematica*

CANDiensten

Amsterdam, The Netherlands

Info: <http://www.candiensten.nl>

October 15–16, 2001

Numerical Methods for Pricing Financial Derivatives

CANDiensten

Amsterdam, The Netherlands

Info: <http://www.candiensten.nl>

November

November 15, 2001

Introduction to *Mathematica*

CANDiensten

Amsterdam, The Netherlands

Info: <http://www.candiensten.nl>

December

December 15–19, 2001

2001 Sixth Asian Technology Conference in Mathematics

RMIT, Australia

Info: <http://atcminc.com/mConferences/ATCM01/index.shtml>

December 19–20, 2001

Mathematica Intermediate Course

CANDiensten

Amsterdam, The Netherlands

Info: <http://www.candiensten.nl>

March

March 18–19, 2002

Workshop on Under- and Overdetermined Systems of Algebra or Differential Equations

Karlsruhe, Germany

Info: <http://iaks-www.ira.uka.de/iaks-calmet/ADE>

Mathematica News

More *Mathematica* news can be found at www.wolfram.com/news.

webMathematica Merges Web and Computational Power of Mathematica

Wolfram Research, Inc. will soon be releasing webMathematica, a revolutionary product that merges the technical computing power of *Mathematica* with the convenience and accessibility of the web.

webMathematica technology can be used by researchers, scientists, and others looking to make their work readily available to colleagues. Schools, universities, and other educational institutions can design sophisticated interactive courseware or supplemental class material that takes advantage of *Mathematica*'s functionality. Companies can rapidly create customized applications without having to install or maintain software on a number of individual machines. Using webMathematica allows companies to significantly reduce training time and costs while, more importantly, enabling those employees not familiar with the *Mathematica* programming language to take advantage of its computational abilities.

Because webMathematica is built on Java servlets, it is compatible with any web server, servlets engine, or application server that supports the Servlet API 2.0 (or higher), giving developers a broad choice of platforms and web solutions from which to work. webMathematica applications can be implemented easily using only standard HTML tools and a working knowledge of *Mathematica*.

webMathematica comes in two license configurations: webMathematica *Professional*, intended for development and deployment of sophisticated intranet and internet sites; and webMathematica *Amateur*, designed for those who wish to showcase their work over the web. Subscribers to Wolfram's *Premier Service* will receive complimentary copies of webMathematica *Amateur*.

For further description, see the product profile of webMathematica in this issue.

Some examples of the use of webMathematica are available at library.wolfram.com/explorations/web.

For more information about webMathematica, see www.wolfram.com/products/webmathematica.

2001 Mathematica Developer Conference Nears

The *Mathematica* Developer Conference, October 11–13, 2001, is an opportunity for those interested in creating *Mathematica*-related tools, products,

courseware, or literature—or for those who wish to offer services as a *Mathematica* trainer, consultant, or evangelist.

The conference will give users a chance to get the newest developer information and will be an excellent opportunity to meet Wolfram Research staff and other *Mathematica* experts from around the world. The conference will feature tutorials, forums, an open computer lab, special events, and many informative, practical sessions.

This year's conference will focus on a variety of topics ranging from user interface enhancements and building blocks for numerical functionality to book and courseware development and application development. Particular emphasis will be placed on web*Mathematica*, MathML, XML, and *J/Link*.

The conference will be held at Hawthorn Suites in Champaign, Illinois. Information about training sessions will be announced as it becomes available at www.wolfram.com/devconf2001.

Mathematica for Mac OS X Announced

Mathematica will soon be available in a version that takes full advantage of the unique capabilities of Mac OS X. The Mach 3.0 kernel and Unix-like foundation of Mac OS X allow this version of *Mathematica* to far surpass older Macintosh versions of *Mathematica* in speed, scalability, and the ability to handle calculations requiring open-ended amounts of memory.

Wolfram Research plans to release *Mathematica* 4.1 for Mac OS X in the third quarter of 2001. Customers who buy *Mathematica* for Macintosh now will be eligible to get the final Mac OS X version free when it is released. Wolfram Research began shipping a preview beta version of *Mathematica* 4.1 for Mac OS X to its Premier Service customers in mid-April.

For more information: www.wolfram.com/news/macosexrelease.html.

Documentation Center Online

An updated Wolfram Research Documentation Center is now online. A collection of reference materials about *Mathematica*, *CalculationCenter*, and related products, the site allows users the means to access software documentation without ever leaving the internet. The address for the Documentation Center is documents.wolfram.com.

International Mathematical Olympiad 2001 Results Site

The 42nd International Mathematical Olympiad (IMO), held July 1–14 in Washington, DC, brought together over 450 of the most talented high-school-

age mathematicians from more than 80 countries. Notable performers from IMO 2001 include USA team member Reid Barton, who became the first contestant ever to win four IMO gold medals. He, along with teammate Gabriel Carroll, received a perfect score on this year's exam. Overall, the USA team received four gold and two silver medals. The China team also repeated its outstanding performance of last year, with all team members taking home gold medals; team Russia was close behind with five gold medals and one silver.

This is the first time the IMO has had a scoring site. The site, hosted by IMO title sponsor Wolfram Research, Inc., contains scoring results for all winning students as well as the competition problems and solutions. Scoring distributions and graphs were generated using *Mathematica*. In addition to hosting the scoring web site, Wolfram Research also donated a copy of *Mathematica* for Students 4.1 to each IMO student contestant.

Also available at the IMO site are contestant photos and profiles, interesting IMO facts (such as the names of Fields Medalists who competed previously), and other pictures from this year's event.

The official IMO scoring site address is imo.wolfram.com.

The Integrator Available in Japanese

A Japanese translation of The Integrator has been made available online. Japanese-capable browsers can find it at integrals.wolfram.com.

Fourth International *Mathematica* Symposium Held

The Fourth International *Mathematica* Symposium (IMS 2001) was held in June at the Chiba Campus of Tokyo Denki University. *Mathematica* notebooks from selected conference presentations are available online at library.wolfram.com/conferences/IMS2001.

***Mathematica* Training Courses to be Held**

Wolfram Education Group offers certified *Mathematica* training to individuals and corporations. October classes are to be held in Denver, Baltimore, Dallas, and Phoenix. Online registration and additional information about Wolfram Education Group are available at www.wolfram.com/services/training/education.

Calculation City: Team Effort from Wolfram and *simplyengineering*

Wolfram Research, Inc. has teamed up with *simplyengineering.com* to provide Calculation City, an online resource that uses web $Mathematica$ to provide engineering professionals with real-world calculators for stress, load, friction, fluid flow, pressure loss, and many other applications.

See www.wolfram.com/news/simplyengineering.html.

Wolfram Web Store Redesigned

The Wolfram Worldwide Web Store has been redesigned with a new time-saving, easy-to-navigate interface. Customers can now purchase directly from participating resellers through the web store, which offers the convenience of using a local currency for payment and a local distributor for shipment. The store's address is store.wolfram.com.

New Products

CalculationCenter™

Based on more than four years of development, *CalculationCenter* is calculation software that combines powerful computational abilities with a simple and intuitive user interface, and an easy tool for professionals who need quick solutions to technical problems.

CalculationCenter has been designed and tested to the same exacting standards as *Mathematica*. It guides the user through every step of solving a problem.

CalculationCenter performs symbolic as well as numerical calculations to solve equations, calculate integrals, manipulate matrices, factor polynomials, analyze data, and so forth.

CalculationCenter has hundreds of functions to handle basic and advanced math, symbolic calculations, matrices, data analysis, graphics, and much more. Its features include basic *Mathematica*, algebra, calculus, matrix and data operations, graphics, solvers, user definitions, units and constants, interface and help features, and document text styles and style sheets.

InstantCalculators are presented for each function to input and automatically set up a problem, while graphics and animations help make calculations visually clear. You can create 2D or 3D plots of functions or data easily. You can choose a range of styles—including bar charts, pie charts, contour plots, and density plots—or let the SmartPlot feature choose an appropriate style.

CalculationCenter saves work in *Mathematica*'s .nb notebook format, which integrates text, graphics, and typeset equations in one document. Notebooks are platform-independent and can be shared with colleagues using the free reader software, *MathReader*, or other Wolfram Research products.

More info

<http://www.wolfram.com/calculationcenter>

Mathematica for Students Version 4.1 Japanese Edition

Mathematica for Students Japanese Edition provides language resources exclusively for Japanese-speaking *Mathematica* users. *Mathematica for Students Japanese Edition* includes a full Japanese user interface and online documentation for *Mathematica*, as well as translations of two of the most well-known and definitive guides to *Mathematica*: *Getting Started with Mathematica* and *The Mathematica Book*, 4th ed., by Stephen Wolfram, the system's creator.

The entire *Mathematica* interface—including menus, palettes, dialog boxes, error and warning messages, and over a thousand pages of online help—has been translated by technical professionals with years of experience with the *Mathematica* system.

With *Mathematica for Students Japanese Edition*, Japanese-speaking students can work faster and more comfortably in their own language.

Mathematica for Students Version 4.1 Japanese Edition was released in April 2001. This popular student version of *Mathematica* is available for Microsoft Windows and Macintosh platforms.

More info

Mathematica for Students: <http://www.wolfram.com/products/student>

Application packages for students: <http://www.wolfram.com/products/student/applications>

UnRisk

UnRisk is a tool designed to combine speed with precision in the analysis of financial derivatives. By integrating *Mathematica* with optimized C++ libraries, *UnRisk* provides the only ready-built solution that offers both the flexibility of a high-level programming language and the speed necessary for production use.

Designed for use by quantitative analysts, traders, risk managers, treasurers, and product designers, *UnRisk* can be used from within the *Mathematica* front end or, with the addition of *Mathematica Link for Excel*, as a spreadsheet add-in.

Key capabilities of *UnRisk* include valuation of equity and interest rate derivatives; calculation of implied volatility, hedge, and sensitivity parameters; sensitivity analysis with respect to market data; insight into complex contracts by means of graphical exploration; rapid modeling, configuration, and parameterization; and “what-if” analysis with respect to contractual rules and potential market developments.

The *UnRisk* application package is developed and supported by UnRisk Consortium, made up of MathConsult GmbH and UNI SOFTWARE PLUS GmbH.

More info

<http://www.wolfram.com/news/unrisk.html>

Operations Research 2.0

SoftAS GmbH has released a new version of *Operations Research*, a *Mathematica* application package that provides the user with tools for handling a broad range of topics in business optimization. With *Operations Research 2.0* and *Mathematica*, you can now solve problems in quadratic programming and combinatorial heuristics, as well as in linear optimization, shortest path tasks, and combinatorial optimization.

Operations Research 2.0 is a collection of four subpackages that can be called from the master package from within *Mathematica*. The documentation includes example notebooks that guide the user through typical problems encountered in university courses and in daily practical work. Each problem is explained, modeled, and solved in full detail, and the complete collection covers a broad class of optimization tasks.

More info

<http://www.wolfram.com/news/operationsres2.html>

The Mathematical Explorer

The Mathematical Explorer is stand-alone software for mathematics hobbyists and people intrigued by mathematics and some of its more challenging topics. It is presented as a set of 15 notebooks, each one covering a different subject, where users can read about the cultural and historical background of each topic and experiment with it.

The software is user-friendly, yet professional, and was conceived as a recreational software (as opposed to a purely educational software or a professional tool software, like *Mathematica*). It is a custom product, based on *Mathematica* technology. It has a small subset of kernel functions enabled, a reduced set of menu items, palettes, documentation and style sheets.

More info

<http://www.wolfram.com/explorer>

PRODUCT PROFILE

webMathematica

webMathematica: Using Mathematica over the Web

Tom Wickham-Jones

Wolfram Research, Inc.

■ Introduction

webMathematica is a new technology that supports the generation of dynamic web content with *Mathematica*. It integrates *Mathematica* with a web server.

□ What is *webMathematica*?

The *webMathematica* product provides tools for setting up *Mathematica*-based web server pages, and a license permitting the use of *Mathematica* over the web. With *webMathematica*, users get the latest version of *J/Link*, *Mathematica* 4.1, and version 1.0 of the *MSPTools*. The *MSPTools* include everything that is necessary to support a *webMathematica* site.

□ Why *Mathematica* on the Web?

Probably the major reason for using *Mathematica* over the web is for computation. The system contains a large collection of functions for calculating in many areas and all of this functionality becomes available over the web. This is particularly important since many web technologies, though powerful in many areas, are not very suited to scientific computation—it is simply not their main focus. *Mathematica* is very suitable for scientific computation, and this is what it can provide for web computations.

Mathematica contains a high-level, interactive, functional programming language. It lends itself to rapid prototyping but can scale up to large intensive computations. These benefits carry over to web content generation where they can significantly reduce the time spent developing the site.

Mathematica has a powerful, though complex, connection technology in *Math-Link*. Now connecting to *Mathematica* has been dramatically simplified with the use of *J/Link*. This technology makes *Mathematica* more suitable for integrating into web computation.

The *Mathematica* front end has long provided the foremost mechanism for working with the *Mathematica* kernel. One might expect that the front end has no place in a web infrastructure. In fact, the front end and a browser give different and complimentary ways to use *Mathematica*. The browser gives a simpler, more interactive interface; people already know how to use browsers so the start-up time is reduced. The front end ultimately gives a more powerful way to work with *Mathematica*. It is possible to combine the two ways of interacting with *Mathematica* so that you can start in a browser and move to working in the front end.

Mathematica is known as a premium system for interactive mathematical typesetting. What is not so well known is that it is a powerful system for working with MathML. MathML is designed to allow the use and reuse of mathematical and scientific content on the web, and by other applications. As MathML becomes increasingly important, it is very advantageous that *Mathematica* work well with MathML.

webMathematica allows the many features and benefits of *Mathematica* to be used over the web.

□ Why a Web Interface?

A web interface brings many of its typical benefits to working with *Mathematica*. For instance, a web browser is the most common type of software that people learn, so people need less training to use a web interface to *Mathematica*. Also, people who develop servers and dynamic web sites will be able to use their favorite tools to develop *Mathematica*-related web sites. This makes the web applications that people build with *Mathematica* easier to build and more powerful. Another major benefit of the web is its ease of delivery. There is no need for installation of special software and access can be controlled and measured with standard technologies.

webMathematica allows the many features and benefits of web technology to be used with *Mathematica*.

□ Potential Areas of Use

Individual scientists and researchers often develop packages for special calculations, which could be done with *Mathematica* or with other computational tools. If they want to give these tools to their colleagues, they often have a problem because their colleagues may not have access to or experience with the necessary components, and so it can be hard to share this computational work. webMathematica solves this problem.

People who develop calculations in *Mathematica* can build sites to deliver interactive calculations over the company web intranet. The people who connect to and use these sites do not have to know anything about *Mathematica*; in fact, they do not need to know they are even using *Mathematica*. In addition, since they are simply using the web, there are no special requirements for installing and administering software on individual machines.

Finally, companies can develop sites that provide computational services to a broader audience than just their own intranets. For example, they can provide custom calculators for special customers. Schools, universities, and other educational establishments can use web $Mathematica$ to deliver sophisticated courses over the web, including highly interactive courseware. Authors and publishers can publish live interactive technical books, supplements, and journals over the web with web $Mathematica$.

web $Mathematica$ provides a way for researchers, scientists, and artists to showcase their $Mathematica$ -based work using interactive web calculators.

■ Dynamic Web Technology

The web is made up of a large collection of servers—programs that wait for requests for material. Requests are sent from clients, which are often operated by users who want to collect information from a server. Typically, a server provides an application that supports the Hyper Text Transfer Protocol (HTTP). Common servers include the Apache Web Server and Microsoft Internet Information Server. Another class of server is application servers; they are large programs that support HTTP as well as connections to databases and other tools. The client software will very often be a web browser such as Internet Explorer, Netscape Navigator, or Mozilla. In order for a web site to support interactive computation, it needs to provide some dynamic technology, which can be supported either by the client or the server.

Several technologies provide dynamic content from a server; examples are CGI scripting, Active Server Pages, and Server Plug-ins. Java Servlets and Java Server Pages are both standard technologies for dynamic web content, available on all platforms, and can be integrated with most web technologies. web $Mathematica$ is based on Java servlets.

A number of client technologies supports dynamic behavior. All of these technologies, including HTML forms, JavaScript, applets, plug-ins, and Active-X controls, can be integrated to work with each other. web $Mathematica$ can work with all of these client technologies.

■ $Mathematica$ Server Pages

$Mathematica$ Server Pages (MSP) technology is the foundation of web $Mathematica$. It is based on a standard Java technology called *servlets*. Servlets are special Java programs that run on a web server machine. Typically, support is provided by a separate program called a *servlet container* (or sometimes a *servlet engine*), which connects to the web server. Essentially, all modern web servers support servlets natively or through a plug-in servlet container. This includes Apache, Microsoft's IIS and PWS, Netscape Enterprise Server, iPlanet, and application servers (such as IBM WebSphere).

MSP technology allows a site to contain HTML pages that are enhanced by the addition of *Mathematica* commands. When a request is made for one of these pages, which are called MSP scripts, *Mathematica* commands are evaluated and the computed result is placed in the page.

□ MSP Scripts

MSP scripts contain HTML that is marked up with *Mathematica* commands. They are easy to write and fit well with HTML development mechanisms such as HTML editors. Here is a sample MSP script.

```
<html>
<head>
  <title>Expanding Polynomials</title>
</head>
<body>
  <h1>Expanding Polynomials</h1>
  <form action="Expand" method="post">
    Enter a polynomial (eg x+y):
    <input type="text" name="expr" size="10">
    Enter a positive integer (eg 4):
    <input type="text" name="num" size="3">
    <br>

    <%Mathlet
      MSPBlock[{$$expr,$$num},
        Expand[$$expr^$$num]] %>

    <br>
    <input type="submit" name="button" value="Evaluate">
  </form>
</body>
</html>
```

This is very much standard HTML except for the Mathlet tag. It contains the *Mathematica* commands that are to be evaluated when the page is requested from the server. Here the input polynomial, referenced by the variable **\$\$expr**, is raised to the power given by **\$\$num**, and expanded. The web page that is sent back to the client contains the computed result. **MSPBlock** is a command provided by webMathematica for working with input sent with an HTTP request. There is a large collection of *Mathematica* commands for working with web requests and responses that take care of features such as plotting, formatting and typesetting, embedding applets, and returning general content.

□ Setting Up

Setting up a web $Mathematica$ site involves first installing and configuring a server that supports Java servlets. When this is running, the web $Mathematica$ tools are then installed on the server. These contain all the material that is necessary to support the site, including the following items.

- A $Mathematica$ application—support for the $Mathematica$ commands
- A $Mathematica$ session manager—software for launching and shutting down $Mathematica$ sessions, as well as support for configuring and monitoring
- A Web Application—an application for loading into the servlet container
- Example MPS scripts
- Sample configuration files
- Documentation
- Applets
- Images

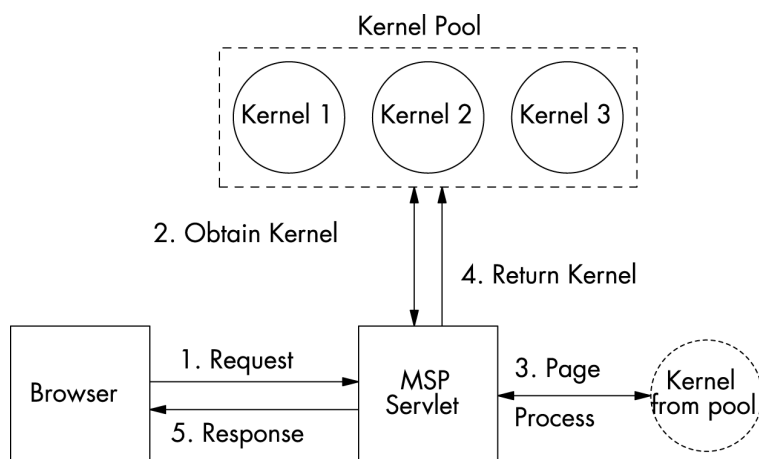
□ Technology Overview

The core of the web $Mathematica$ site is the MSP servlet. It uses the $Mathematica$ session manager, a useful application that maintains a pool of $Mathematica$ sessions that are waiting to be used. When a request is received, the MSP servlet obtains a $Mathematica$ kernel and uses it to process the MSP script. When this is finished, the kernel is returned to the pool and the result returned to the browser. A given $Mathematica$ session is reused many times, which leads to an improvement in performance. This arrangement allows one server to work with more than one $Mathematica$ kernel, allowing concurrent computations.

The result of a request to a web $Mathematica$ site can be anything that can be generated by $Mathematica$: this includes HTML, MathML, XML, images, and $Mathematica$ notebooks. The results can interoperate with applets, javascript, plug-ins, and Active-X controls.

The $Mathematica$ session manager takes care of launching and initializing $Mathematica$ kernels. It also takes care of shutting down sessions that exceed a preset limit for computations. The whole system is fully configurable to provide detailed control of a particular site.

An overview of the steps in processing a page is shown below. For the sake of illustration, it has a $Mathematica$ session pool with three sessions.



■ Summary

webMathematica allows a web server to generate dynamic content with *Mathematica*. It is driven by a technology called *Mathematica Server Pages*. MSP technology works by placing MSP scripts, HTML pages that contain *Mathematica* commands, on your server. When a request for a page is received, these commands are evaluated by *Mathematica* and the result is returned. MSP technology is based on Java servlets, a standard and fully supported Web technology. They are fully integrated with many client and server web technologies.

Wolfram Research provides the necessary tools for a Java servlet-enabled web server. The person or company wanting to host a webMathematica site must provide the server.

More information on webMathematica can be found at www.wolfram.com/products/webmathematica, and examples of webMathematica can be found at library.wolfram.com/explorations.

New on MathSource[®]

The following notebooks and packages were recently published on *MathSource*. To download, select the name of any of these items or visit *MathSource* at www.mathsource.com/Content/WhatsNew.

- *Generating Feynman Graphs and Amplitudes with FeynArts* (Thomas Hahn, item 0202-194)
- *Dot-Dash Scatter Plots* (Eric Gossett, item 0203-263)
- *MultiplierMethod—A General Purpose Nonlinear Programming Algorithm for Constrained Nonlinear Optimization* (Jean-Christophe Culioli, Joseph P. Skudlarek, item 0207-289)
- *Programming Paradigms via Mathematica (A First Course)* (Richard D. Neidinger, John R. Swallow, Todd G. Will, Stephen L. Davis, item 0208-583)
- *Integrability Package: Computation of Invariants and Symmetries* (Unal Goktas, Willy Hereman, item 0208-932)
- *Analysis and Design of Electronic Circuits with Analog Insydes* (Thomas Halfmann, Jutta Praetorius, Tim Wichmann, item 0209-731)
- *Graphica 2 and Beyond: Creating Artistic Images with Artlandia* (Artlandia, Inc., item 0209-966)
- *LoanFunctions.m Package* (David Reiss, item 0210-856)
- *qb-math: A MathLink Interface to Qbull's Delaunay Triangulation* (P.J. Hinton, Alban Tsui, item 0211-251)
- *Finite Element Method* (Andreas Stahel, item 0211-262)
- *Solving Two Linear Equations with Web-Based Plot* (Youngcook Jun, item 0211-273)
- *Experiments with Finite Element Method* (Edgar Camacho Cota, item 0211-284)
- *Creeping Flow Past a Stationary Sphere* (Mark J. McCready, item 0211-295)
- *Partitions and Sums of Integer Powers* (Jaime Rangel-Mondragon, item 0211-307)
- *Baseball Trajectory with Graphics* (Robert Lee, item 0211-318)
- *When Close Enough Is Close Enough* (Hu Zhe, item 0211-329)
- *A Practical Guide to the Propagation and Analysis of Experimental Error* (Anthony Muscat, item 0211-330)

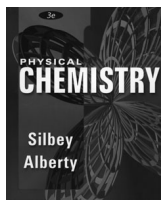
- *Gluing Two Moebius Strips into a Klein Bottle* (Srdjan Vukmirovic, item 0211-341)
- *Automatic Analytical Computation of Feynman Diagram Expressions* (Alexander A. Tetervak, item 0211-352)
- *Vectorial Analysis* (Renan Cabrera, item 0211-363)
- *Cuantica: A Package for Quantum Mechanical Calculus* (Renan Cabrera, item 0211-374)
- *KeyFraming in Animation* (Louis J. Nachman, item 0211-385)
- *A RealOnly Package That Can Be Disabled or Enabled* (Ted Ersek, item 0211-396)
- *Interference and Diffraction Patterns* (Renan Cabrera, item 0211-408)
- *Tensorial: A Tensor Calculus Package* (Renan Cabrera, item 0211-419)
- *Variational Calculus Examples* (Renan Cabrera, item 0211-420)
- *The Mathematica Journal Vol. 6, No. 4—Electronic Supplement* (Wolfram Research, Inc., item 0211-431)
- *Famous Fractals* (Renan Cabrera, item 0211-442)
- *Basic Examples of Relativity Theory* (Renan Cabrera, item 0211-453)
- *Integral Equation Solver* (Ho Woong Choi, item 0211-464)
- *Basics of Non-Linear Dynamics* (Renan Cabrera, item 0211-475)
- *Extending Quaternions to Rotations* (Peter Meulbroek, item 0211-486)
- *Basic Plane Computational Geometry* (Adriano Pascoletti, item 0211-497)
- *TangMath* (Franco Cocchini, item 0211-509)
- *Context-Free Languages: Chomsky Normal Form* (Jaime Rangel-Mondragon, item 0211-510)
- *A Catalog of Cellular Automata* (Jaime Rangel-Mondragon, item 0211-521)
- *Functions EvaluateAt, EvaluateAtPattern, PositionMin, PositionMax, and More* (Ted Ersek, item 0211-532)
- *Logic Programming* (Renan Cabrera, item 0211-543)
- *Humorap 1.0: A Human Model for Radiological Protection (ICRP 66 and 78)* (Guillermo Sanchez, item 0211-554)
- *Solution of Two-Point Boundary Value Problems: A Root-Solving Approach* (Robert D. Tzou, item 0211-565)
- *A Package for PolarListPlots with Polar Gridlines* (Ted Ersek, item 0211-576)
- *A Function Called HoldTemporary* (Ted Ersek, item 0211-587)

- *Automatic Printing of Multiple Mathematica Files* (Roeland Merks, item 0211-598)
- *Compartmental Modeling* (Guillermo Sanchez, item 0211-600)
- *MomCumConvert: A Package for Conversion Between Moments and Cumulants* (Qi Zheng, item 0211-611)
- *Basic Data for Biochemistry* (Robert A. Alberty, item 0211-622)

New Publications

Physical Chemistry

Robert J. Silbey and Robert A. Alberty, 2001, John Wiley & Sons, Inc., 969 pp., hardbound.



Physical Chemistry offers the basic academic theory of physical chemistry. The text contains reference appendices and sections on thermodynamics, quantum chemistry, statistical mechanics and kinetics, and macroscopic and microscopic structures. Included in the book are many practical problems with applications from physical chemistry, two sets of chapter problems, computer problems for use with *Mathematica*, as well as hand-held calculator problems. The solutions manual contains answers to the calculator problems. *Mathematica* notebooks supply the solutions for each of the 136 computer problems.

Chapters include: Zeroth Law of Thermodynamics and Equations of State; First Law of Thermodynamics; Second and Third Laws of Thermodynamics; Fundamental Equations of Thermodynamics; Chemical Equilibrium; Phase Equilibrium; Electrochemical Equilibrium; Ionic Equilibria and Biochemical Reactions; Quantum Theory; Atomic Structure; Molecular Electronic Structure; Symmetry; Rotational and Vibrational Spectroscopy; Electronic Spectroscopy of Molecules; Magnetic Resonance Spectroscopy; Statistical Mechanics; Kinetic Theory of Gases; Experimental Kinetics and Gas Reactions; Chemical Dynamics and Photochemistry; Kinetics in the Liquid Phase; Macromolecules; Electric and Magnetic Properties of Molecules; Solid-State Chemistry; Surface Dynamics; plus appendix and index.

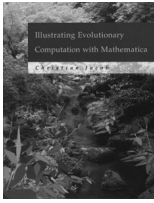
A solutions manual of 229 pages (paperback) is also available.

Available:

<http://store.wolfram.com/view/book/ISBN0471383112.str>

Illustrating Evolutionary Computation with *Mathematica*

Christian Jacob, 2001, Morgan Kaufmann Publishers, 578 pp., hardbound.



Illustrating Evolutionary Computation with Mathematica introduces evolutionary computation and explores the application of evolution to developmental processes in nature. The text introduces major mechanisms of biological evolution and explains genetic algorithms, genetic programming, evolutionary programming, and evolutionary strategies. Included are a detailed explanation of computer-modeled developmental programs and *Mathematica* notebooks for all programs in the book.

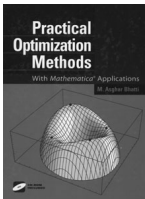
Chapters include: The Fascination of Evolution; Evolutionary Algorithms for Optimization; Genetic Algorithms; Evolution Strategies; Programming by Evolution; Evolutionary Programming; Genetic Programming; Advanced Genetic Programming at Work; Computer Models of Developmental Programs; Evolutionary Inference of Lindenmayer Systems; and Artificial Plant Evolution.

Available:

<http://store.wolfram.com/view/book/ISBN1558606378.str>

Practical Optimization Methods with *Mathematica* Applications

M. Asghar Bhatti, 2000, TELOS/Springer Verlag, 715 pp., hardbound, CD-ROM.



Practical Optimization Methods presents basic optimization theory and computational algorithms in an informal style for undergraduate and graduate students in engineering, operations research, and management information systems. This book uses all of *Mathematica*'s symbolic, numerical, and graphical capabilities to develop understanding of optimization algorithms. *Mathematica*'s graphical capabilities allows for visual comparison of different optimization algorithms. The accompanying CD-ROM contains the implementation of all *Mathematica* functions discussed.

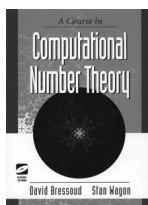
Chapters include: Optimization Problem Formulation; Graphical Optimization; Mathematical Preliminaries; Optimality Conditions; Unconstrained Problems; Linear Programming; Interior Point Methods; Quadratic Programming; Constrained Nonlinear Problems; and with an appendix, An Introduction to *Mathematica*.

Available:

<http://store.wolfram.com/view/book/ISBN0387986316.str>

A Course in Computational Number Theory

David Bressoud and Stan Wagon, 2000, Key College Publishing, 367 pp., hardbound, CD-ROM.



A Course in Computational Number Theory is an introduction to number theory, with emphasis on a broad range of computational issues. *Mathematica* files (on CD-ROM) with all the commands and programs are included as a tool for motivation and explanation. This book covers the theoretical structure of number theory, but expands with the software into exploration of many topics including factoring algorithms, primality testing, and the RSA public-key cryptosystem. Unusual and advanced topics also covered include check-digit schemes, continued fractions, Pell's equation, and the Gaussian primes.

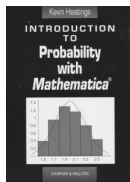
Chapters include: Fundamentals; Congruences, Equations, and Powers; Euler's Function; Prime Numbers; Some Applications; Quadratic Residues; Continued Fractions; Prime Testing with Lucas Sequences; Prime Imaginaries and Imaginary Primes; and appendices of *Mathematica* Basics and Lucas Certificates Exist.

Available:

<http://store.wolfram.com/view/book/ISBN1930190107.str>

Introduction to Probability with *Mathematica*

Kevin Hastings, 2001, Chapman & Hall, 380 pp., hardbound.



Introduction to Probability with Mathematica is designed as an introductory text for probability theory. By using *Mathematica* as the supporting technology rather than as a statistics package, the author focuses on probability concepts and simulation, rather than on statistical analysis. Each section of the book exists as a *Mathematica* notebook on the CRC web site. Emphasis is placed on sampling distribution and statistical inference, allowing readers to see probability as integral to data analysis.

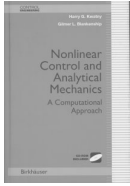
Chapters include: Discrete Probability; Discrete Distributions; Continuous Probability; Continuous Distributions; Asymptotic Theory; Applications of Probability; Appendix.

Available:

<http://store.wolfram.com/view/book/ISBN1584881097.str>

Nonlinear Control and Analytical Mechanics: A Computational Approach

Harry G. Kwatny and Gilmer L. Blankenship, 2000, Birkhauser, 317 pp., hardbound.



Nonlinear Control and Analytical Mechanics: A Computational Approach is a resource for mechanical and control engineers at all levels, from graduates to professionals. This book highlights and utilizes the computational infrastructure common to both modern analytical mechanics and nonlinear control. An integrated approach gives a set of models and control design examples that are contemporary and practical. There are instructions for building explicit mathematical models for engineering design as well as simplified models for easier use. A CD-ROM provides extensive examples and exercises using *Mathematica* notebooks and programs.

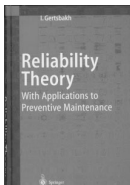
Chapters include: Introduction; Introduction to Dynamical Systems; Introduction to Differential Geometry; Kinematics of Tree Structures; Dynamics; Smooth Affine Control Systems; Robust and Adaptive Control Systems; Variable Structure Control; Appendix with Quick Reference Tables; Index.

Available:

<http://store.wolfram.com/view/book/ISBN0817641475.str>

Reliability Theory with Applications to Preventive Maintenance

Ilya Gertsbakh, 2000, Springer-Verlag, 219 pp., hardbound.



Reliability Theory with Applications to Preventive Maintenance is a textbook on reliability theory that is appropriate for graduate students, researchers and engineers specializing in quality control, logistics, reliability, and maintenance engineering. The text focuses on applications in preventive maintenance (PM). All models are presented in connection with the relevant statistical material. Each chapter is concluded by a series of exercises with detailed solutions using *Mathematica*. Topics discussed are PM with learning, choice of the best time scale for PM, handling multidimensional state description, and dealing with uncertainty in data.

Chapters include: System Reliability as a Function of Component Reliability; Parametric Lifetime Distributions; Statistical inference from Incomplete Data; Preventive Maintenance Models Based on the Lifetime Distribution; Preventive Maintenance Based on Parameter Control; Best Time Scale for Age Replacement; Preventive Maintenance with Learning; Solutions to Exercises; Appendix A: Nonhomogeneous Poisson Process; Appendix B: Covariances and Means of

Order Statistics; Appendix C: The Laplace Transform; Appendix D: Probability Paper; Appendix E: Renewal Function.

Available:

<http://store.wolfram.com/view/book/ISBN3540672753.str>

Editor's Introduction

Veikko Keränen, Guest Editor

Principal Lecturer, Rovaniemi Polytechnic, Rovaniemi, Finland

It is a delight for me to see this *IMS '99* special issue of *The Mathematica Journal* in preparation just before the next International *Mathematica* Symposium takes place at the Chiba Campus of Tokyo Denki University, Japan, June 25–27, 2001. My warmest thanks go to Joe Grohens and Flip Phillips for assembling the present issue.

IMS '99 took place at the Research Institute for Symbolic Computation, Hagenberg, Austria, August 23–25, 1999, under the leadership of Bruno Buchberger whose keynote talk is now being published (in synopsis) for the first time. The participants for *IMS '99* came from five continents, and this fact makes *IMS* a truly international event.

In contrast to the previous symposia, we placed the full *IMS '99* proceedings on the web at south.rotol.ramk.fi/~keranen/IMS99. Indeed, the electronic format is very convenient for interactive material and has no size limitations. Even so, this issue of *The Mathematica Journal* definitely enables a wider distribution and awareness of the work for *IMS '99*.

I would like to conclude with a short history of the *IMS*, its sister organization, International Arctic Seminars (*IAS*), and look a bit ahead into the future.

In my mind, the beginning of the International *Mathematica* Symposia dates back to 1992 when Wolfram Research organized *Mathematica* Days in Rotterdam, the Netherlands. Among the participants were Peter Mitic, Gautam Dasgupta, Pertti N  ykki, Klaus Sutner, and Robert Kragler. Although most of us met for the first time and came from quite different walks of life, it was clear that we shared the same enthusiasm for algorithmic mathematics. Through a very spontaneous and natural process, we started to organize hands-on training, first in Rovaniemi, then in Murmansk, and later in many other places. A part of these events can be found on the web site south.rotol.ramk.fi/IMS_IAS_Events.html.

It was at the end of 1993 when Peter Mitic introduced the idea of a series of symposia devoted to applications of *Mathematica* in the various fields of education, research, and art. The name, International *Mathematica* Symposium, was then suggested by Stephen Wolfram at the 1994 *Mathematica* Developer Conference (Champaign, Illinois, USA), when the first announcement was made for *IMS '95* to be held in Southampton, England. In Southampton we really could sense the atmosphere of a *Mathematica* family. Thus, *IMS* had been successful in providing a forum for people from different fields to interact.

More and more people were joining in the process of organizing the symposia. In 1996, the *LAS* started its work at Murmansk State Pedagogical Institute in Murmansk, Russia, in collaboration with Murmansk State Technical University. Nowadays, Robert Kragler is the chair of these *LAS* events.

At *LAS* '97 in Murmansk, we met Ronald Rusay, whose paper, jointly authored with Erik Jensen, now appears in this issue of *The Mathematica Journal*. Just after returning from *LAS* '97 to Rovaniemi, Finland, together with Gautam Dasgupta and Klaus Sutner, we started *IMS* '97 in Rovaniemi, where Antero Hietamäki, Peter Mitic, and many others had been making last-minute preparations for the event. *IMS* '97 contained a separate two-day event for high schoolers in addition to pre- and post-conference tutorials and workshops on advanced theoretical topics. More on the history of *IMS* '99 is covered in this issue.

We look forward to *IMS* 2001, which will take place very soon at Tokyo Denki University, Japan, under the leadership of Yoshihiko Tazawa. I'm sure that it will be an exciting and enjoyable event.

Regarding the future, I would like to mention the active role of Phil Ramsden, Imperial College, UK, and Christian Jacob, University of Calgary, Canada, in the process of organizing the forthcoming *IMS* events.

There are still a great many more people, too numerous to be mentioned here, who have really been invaluable to the development of *IMS*. They come from inside and outside of Wolfram Research—and even outside the lists of participants. I wish to thank them all and extend the thanks to Wolfram Research who has provided substantial resources to support, for instance, a number of keynote speakers and tutors for *IMS* events.

Finally, I gratefully acknowledge the recent support of the Borealis Software Engineering Programme as well as the long-standing support of the School of Technology at the Rovaniemi Polytechnic for promoting the *IMS*.

The Foundations of Mathematics and Mathematica

Transcript of keynote video-conference address, August 23, 1999

Stephen Wolfram

Wolfram Research, Inc.

www.stephenwolfram.com

Note: This talk represents the state of Stephen Wolfram's thinking on these topics at the time of *IMS '99*. Since then he has made considerable further progress, which will be described (along with a very great many other things) in his forthcoming book *A New Kind of Science*.

Well I'm very happy to be here today, if only in virtual form.

I guess when this conference was first being planned, I had hoped very much to be able to come also in physical form.

But as some of you probably know, for the last eight years I've been doing a very very big science project, and to get it finished, particularly recently, I've had to become more and more of a recluse.

In fact you would probably be surprised at how little I'm getting out now: in fact the very last conference I traveled any distance to was the IMS in Southampton four years ago.

So anyway I'm very much looking forward to having my science project finished, and being kind of free again.

And it's perhaps because I've been a recluse so long I've decided that here I would try to talk about some somewhat advanced and abstract things—about the foundations of mathematics, and their relationship to my big science project, and their relationship to *Mathematica*.

Originally presented at the Third International *Mathematica* Symposium (Hagenberg, Austria, August 23–25, 1999).

Some of you probably know a little about what my science project is about—though I haven’t talked much about it. What I’m doing is something very ambitious.

I’m basically trying to build a whole new kind of science, certainly as big as any of the existing sciences, like physics and chemistry and things, and perhaps in some ways bigger.

The basic point of the science I’m trying to build is this. If you look at the history of science for the last 300 years, it’s probably fair to say the hard sciences have been following one basic idea: to find mathematical equations that represent things in nature.

Well, that idea worked pretty well for Newton and friends in studying orbits of planets and things like that. But in lots of other cases—for example in biology—it’s really never worked at all.

So the question is: what else can one do?

Well, I think it’s reasonable to assume that nature follows definite laws, definite rules: otherwise we couldn’t do science at all. But the question is: why should those rules be based on the constructs that we happen to have come up with in traditional mathematics?

Why should they be about numbers, and derivatives, and all those kinds of things in mathematics?

Well, what I decided to try to do nearly 20 years ago now was to see what science would be like if one thought about using more general kinds of rules: the kinds of rules that can easily be embodied in computer programs, but can’t necessarily be represented easily in traditional mathematics.

Some of you probably know some of the things I did in the early 1980s on cellular automata, and getting the field of complex systems research started. And I’m pretty happy with the work I did then, though I’m actually far from thrilled with the way the field developed in general terms afterwards.

I figured out quite a lot of stuff in the early 1980s: enough to convince me that the idea of generalizing the kinds of rules one uses to study nature isn’t completely crazy.

But then in the mid-1980s I got kind of stuck—I didn’t have very good tools to use for my computer experiments, and I seemed to be spending all my time writing little pieces of code and gluing them together and so on. And it was about then that I had the idea that perhaps I should build a big software system that would be able to do all the things I needed, and might even be useful to some other people as well.

And that was kind of one part of how *Mathematica* came to be. Of course since I’m a practical fellow, I tried to design *Mathematica* to be as useful as possible to as many other people as possible, too.

But a big part of my motivation for building *Mathematica* was that I wanted to use it myself.

And I'm happy to say that starting just after Version 2 came out, I was able to start doing that very seriously.

Well, I've discovered a huge huge amount of science with *Mathematica*. And I'm very much looking forward to telling everyone about it. But it's a big intellectual structure that I've been building, and I'm not quite ready to talk about all of it yet.

I'll talk about a few pieces here. And I hope that when my book about all of this—it's called *A New Kind of Science*—is out you'll all have a chance to really read about it more completely.

Well, what I want to do today is actually to talk about a topic that's sort of at the intersection of my two most favorite kinds of topics: my new kind of science and *Mathematica*.

It turns out that at that intersection are questions about the foundations of mathematics, and there are some new things I've realized about the foundations of mathematics from working on my new science, and also from working on *Mathematica*.

I'm particularly happy to be talking about this here because there are several people in the *Mathematica* community who I've really enjoyed interacting with on these topics—though they definitely don't always agree with me—particularly including Bruno Buchberger, Dana Scott, Klaus Sutner, and Greg Chaitin.

OK, well what I'm going to say here may be somewhat abstract, but I hope and believe that with all of your background in *Mathematica*, you'll be almost uniquely in a position to really get something out of what I'm trying to say. So let me try to get started.

I'll tell you a little bit about my new kind of science, then I'll tell you how it relates to the foundations of mathematics, and *Mathematica*.

There are some pretty big shifts in intuition involved in my new science, and I certainly won't be able to explain all of it here. But to understand the rest of what I'm going to say, I have to spend a few minutes explaining some of what my science is about.

Well, as I said, what my science is based on is thinking about what arbitrary sets of rules do: in effect, what arbitrary simple computer programs do.

Now normally when one builds computer programs, one sets up the computer programs to do specific things. But what I'm interested in in my new science is what arbitrary computer programs—say ones one might choose at random—do.

It's very hard—in a sense provably impossible—just to figure this out by pure thought.

But it's easy to do experiments. And what's actually particularly fun about these experiments is that they're so easy that they could even have been done probably thousands of years ago.

Maybe they were. I actually don't think so. Because I think if people had ever seen the results they give then science would have actually developed along a very different track than the one it's actually followed.

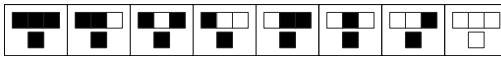
OK, so let's take a look at these experiments, and let's look at what some simple programs do.

I'm going to talk first of all about some things called cellular automata, because they happened to be the first things I looked at in the early 1980s. But one of the things I've discovered in the last eight years is that nothing I'm saying is really in the slightest bit specific to cellular automata.

OK, so what is a cellular automaton?

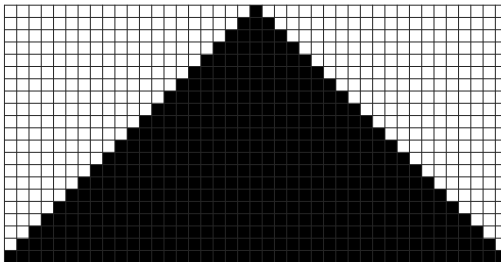
One can set one up by having a line of cells, each one let's say black or white.

And then one has a rule for evolving the thing down the page. So let me show you an example.



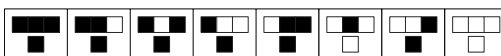
So this might be a rule for the cellular automaton. It says if you have a cell, then that cell will become black on the next step unless it and its immediate neighbors are all white.

So let's ask what happens if we take this particular rule and just start it off from a single black cell.

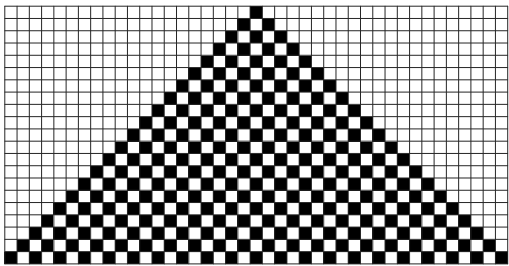


OK, so this is what we get: it's very straightforward. We start off with a single black cell at the top there, we apply this rule over and over again, and we just get a simple uniform pattern.

OK, well let's try changing that rule a little bit and see what happens to the kind of pattern we get. Here's a rule, very similar to the previous one, yet slightly different.



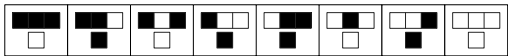
Let's see what that rule does.



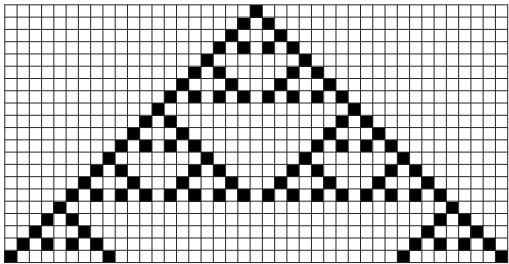
So, what we see is a checkerboard pattern.

Well, at this point we might guess that there's some kind of theorem. And the theorem might say if you have a sufficiently simple rule, and you start it off with a sufficiently simple initial condition, then what you'll get is a simple, let's say periodic or repetitive, pattern.

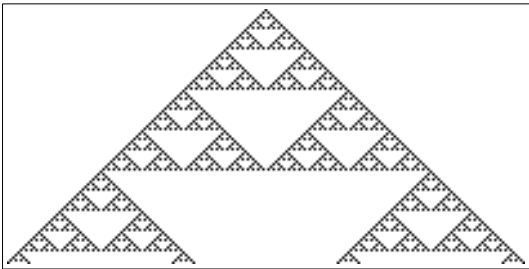
Well, let's try another rule. We can just change the rule we've been using a little bit.



Let's try this rule here. Let's see what pattern that produces.



Well, that produces a rather different pattern. What's this pattern going to do? Let's run it, let's say for 100 steps. Here's the result that we get.



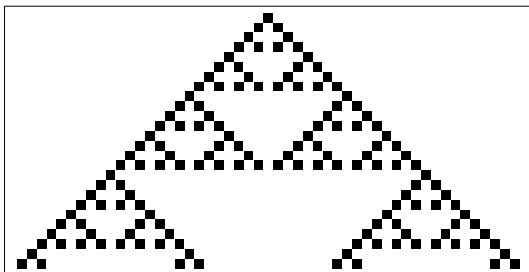
And what we see here is that this particular cellular automata rule, when it starts off from a single black cell, instead of just making a repetitive pattern, it makes a nested self-similar pattern. This is one of those Sierpinski gasket fractals.

Well, just to emphasize how simple a kind of program a cellular automaton actually is, we could for example just write a cellular automaton program in *Mathematica*. Let me show you what a step of cellular automaton evolution looks like in a modern *Mathematica*. Here it is.

```
Sign[BitAnd[2^ListConvolve[{1, 2, 4}, a, 2], num]] &
```

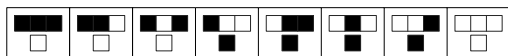
So here for example, `a` is the current state, and `num` is the rule number. So here for instance, this was rule number 90. Let me regenerate the picture that I had above, by just running this little `ListConvolve` thing, which is the cellular automaton rule.

```
Show[Graphics[Raster[1 - Reverse[
  NestList[Sign[BitAnd[2^ListConvolve[{1, 2, 4}, #, 2], 90]] &,
  IntegerDigits[2^25, 2, 51], 25]]],
  AspectRatio -> Automatic, Frame -> True, FrameTicks -> None];
```

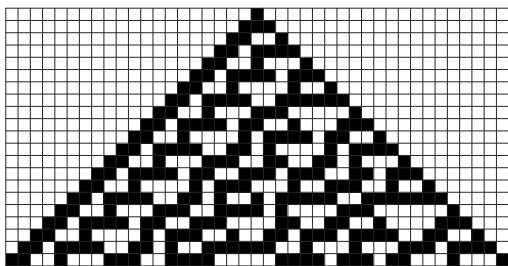


Looking at what we get here, with our simple cellular automaton rule, starting off from a simple initial condition, we again get a pattern that we can recognize as quite simple. It's a nested pattern. So again we might guess that the true theorem is if you start off with a simple rule and a simple initial condition, then you either get a repetitive pattern or a nested pattern. But we might guess—and certainly that was my original guess—that if one has a sufficiently simple setup like that then there's nothing more complicated that one can get. □

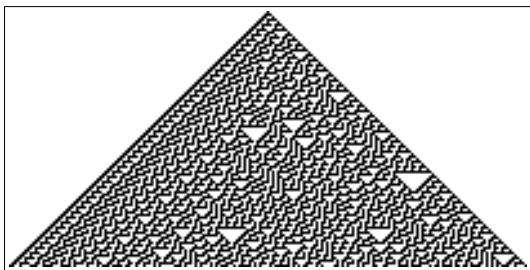
OK, well let's try this particular creature here. Here's another cellular automaton rule, same kind of idea as the previous ones but different particular choices of outcomes for different configurations. □



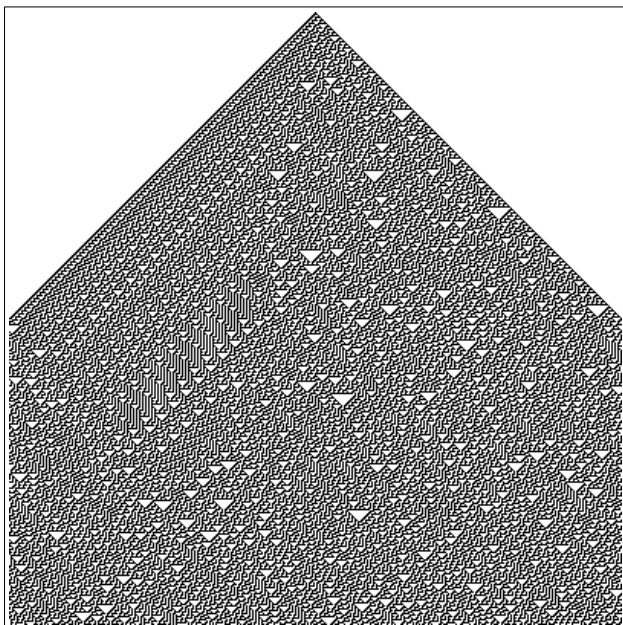
Let's see what this does. □



Well this does something very strange. We're starting off from just a single black square here, but now instead of making something that looks like it's obviously a repetitive pattern or a nested pattern, it looks like some more complicated mess. Well, let's go on. We can run that rule for a bit longer, see what it does.



It doesn't seem to do anything terribly simple. We can keep running it. Let's keep running it, let's say for 400 steps. Here's the pattern that we get.



Well you can see there's some regularity in all this. But much of it is pretty random. And actually I know rather well that it's random, because essentially we've been using the center column of this pattern as the thing that makes integer random numbers in *Mathematica* for the last 11 years. And so a lot of people have tested it, and nobody's ever found any deviation from randomness.

But let's just look at this picture.

It's just nothing like we would expect.

I mean, from building things ourselves and doing engineering and so on, we're used to the idea that if we want to get something complicated we'd better have a complicated set of plans or rules—or else we'll just get something very simple out. But here we have some simple rules, there they are, and we start off from this very simple initial condition, yet we get something very complicated out.

So the question is: what's going on here?

Well, I think what's going on is that we're basically seeing in a very direct form what I think is essentially what has been sort of the big secret of nature for a really long time.

You see, there's a funny thing. If you look at two objects: one's an artifact, one's something made by nature, it's a pretty good heuristic that the one that looks more complicated is the one that nature made. And the one that looks simpler is the one that humans made.

And I suppose it's kind of that sort of thing that made people assume that there had to be some kind of superhuman supernatural intelligence behind the things that got built in nature.

But one of the things that's happened in the past few hundred years is that it's been pretty much figured out what a lot of the underlying rules in nature are. And actually they're really pretty simple.

So that makes it even more bizarre that nature can produce all the complicated stuff we see.

And it's been a big mystery how that can possibly happen.

But I think—and actually now I've accumulated a huge amount of evidence for it—that this thing that I've seen in simple programs like the cellular automaton that's called rule 30, is the key to what's going on.

See, the rules that we use for engineering are essentially special ones that we've set up to be able to do the tasks we want to do. And to make that work these rules have to produce only fairly simple behavior—behavior whose outcome we can reasonably foresee.

Otherwise we wouldn't be able to use those rules to achieve particular tasks. But the point is that nature doesn't have any constraint like that. So it can effectively pick its rules much more at random.

And the big fact that I've discovered is that if one does that a lot of the rules one sees—even if they themselves are very simple indeed—can produce incredibly complicated behavior.

And I think that's basically what the sort of underlying secret of nature is. There are a huge number of questions in all sorts of areas of science that one can tackle for real once one knows this. And that's part of what I've been doing for the past eight years.

But this phenomenon is also important for thinking about mathematics. In fact, I think the general approach I've been taking should lead not only to a new kind of science, but also to something that's essentially a major generalization of mathematics. And that's a large part of what I want to talk about here.

Well let me explain why that is.

One can think of mathematics as being something that effectively tries to work out the consequences of abstract sets of rules.

The question is what kinds of rules one uses.

And I think what's happened is that mathematics has ended up using only rather special kinds of rules.

But in fact there's a vast universe of other rules out there, which mathematics has essentially never looked at.

And actually the character of what those rules do, and what one can do with those rules, is somewhat different from what happens with the ones that are usually dealt with in mathematics.

I should say that if one wants to, one can certainly write a cellular automaton like the one I was showing there, in sort of mathematical form. This is in fact its rule. If P, Q, and R are the colors of neighboring sites, this is the rule for the color of the next site.

$$\text{Mod}[p + q + r + q\,r, 2]$$

But it turns out that even though one might be able to state this as a mathematical rule, it's not a terribly natural thing to do. And the usual ideas of, for example, algebra that one might try to apply to something like this, tell one almost nothing about the behavior that one gets from the rule.

So effectively what's going on here is something that ends up being quite disconnected from the kinds of things that we usually think of as mathematics.

OK, so one question is, if mathematics isn't dealing with genuinely arbitrary abstract rules, what exactly is it dealing with?

I think in the end to figure this out, one effectively has to look at history.

I think defining mathematics is a little like defining life. One knows definitions of life that say something is living if it can move itself around, or if it can reproduce itself, or if it eats things, and so on. But as one goes on and looks at all these definitions, one realizes that there are lots of devices, physical processes, and so

on, that satisfy all of these various definitions, just like life does. And in the end the only definitions that actually work and correspond to what we intuitively think of as being a living system, are ones that involve the specifics of containing DNA and things like that. And essentially involve the particular history that life has taken on Earth over the course of geological time.

And I think it's sort of the same with mathematics. One can come up with all sorts of abstractions for what mathematics might be, but in the end one basically has to look at history.

And essentially one has to go back to things that were happening in ancient Babylon.

What seems to have happened in Babylon is that two things were developed: essentially arithmetic, and geometry.

And I think that what's happened is that since then essentially all the mathematics that's been developed has somehow been derived from those original forms of arithmetic and geometry.

Actually if one looks up the definition of mathematics in most dictionaries, one will find a definition that basically agrees with this.

But it's not the definition that most professional pure mathematicians imagine is the appropriate one for mathematics. They think that somehow mathematics is about studying quite arbitrary abstract systems.

And that's actually how I thought about mathematics when I named *Mathematica* "*Mathematica*".

But if you actually look at what mathematics as mathematicians actually do it is—how that's defined, it's somewhat different.

See for a long time in the history of the development of mathematics, people thought of mathematics as somehow being about describing the way our universe works. And that was the justification that was used for the kinds of constructs that were developed in mathematics, the ones that should be considered, and the ones that should not be considered.

And that's what led for example historically to the fact that the idea of having more than three dimensions to space was a hard thing to introduce, and things like that.

And—particularly after Newton and friends did their stuff—it was kind of felt that for sure the generalizations of arithmetic and geometry that have been so successful in their efforts to kind of put a formalism into science should be enough to describe everything that's relevant in our universe.

But one thing I can tell you pretty definitively from the science that I've done is that that's just not true.

And in fact it's that assumption that's made a lot of science get stuck.

A big part of what I've discovered is that the kinds of rules that nature really seems to follow are ones that are pretty easy to represent in simple computer

programs, but almost impossible to represent in traditional kind of arithmetic-and-geometry mathematics.

So the idea that one shouldn't make the rules in mathematics more general because that would mean going beyond anything that's in our actual universe is just wrong. That's not a reason to not think about generalizing the kind of arithmetic and geometry-based rules in mathematics.

Of course, people in mathematics kind of thought they stopped worrying about making rules correspond to our universe about a century ago.

That was one of the big achievements of Cantor, and to some extent Galois, and friends. To start making constructs in mathematics that were purely abstract, and not intended to be linked to anything in the actual universe.

But OK, without the constraint of linking things to what happens in the actual universe, how do people pick the constructs to use in mathematics?

Well that's an interesting story and it's sort of central to what mathematics is, and what it isn't.

And it turns out that what it all revolves around is the idea that mathematicians, at least in the last century or so, seem to have been proudest of: the idea of proof in mathematics.

I guess that mathematicians always feel that they want certainty; they want to be sure that they are figuring out things that are true.

And they have the idea that the way that one does this always has to do with proofs.

I guess that idea started with Euclid and friends. Euclid didn't want to rely on how some geometrical figure was drawn in the sand or whatever, and whether two angles that looked like they were the same, were the same as far as he could measure them. He wanted to know precisely were these two angles the same, not just did they look the same as well as he could measure them.

He wanted a proof, some kind of logical proof, that these two angles were exactly the same.

So he got into the idea of essentially using logic to take some statement and see if it could logically be deduced from some set of axioms.

Now, to be fair, not all mathematics has historically been done that way. Particularly in areas like number theory, people—people like Gauss for example—quite often did experiments to see what was true.

But particularly as things got more hairy with infinitesimals, and things where everyday intuition seemed never to work, it got pretty much taken for granted that proof was the only way to move forward in mathematics.

And in fact if you look at the Bourbaki books, for example, their opening words are: "Since the time of the Greeks, to say 'mathematics' has meant the same as to say 'proof'".

Well, I myself have never been a great fan of the idea of proof.

And perhaps that shows that I really am a scientist, not a mathematician. I like starting out without knowing what's true, and making observations and doing experiments, and then finding out what's true. Sort of discovering things from scratch.

I don't like somehow having to guess what's true, then work backwards and try to find a proof of it.

And actually my experiences have kind of strongly conditioned me this way by now. Because in the science I've done I've discovered—by being I think a fairly careful experimenter—things that I'd never possibly have guessed.

And actually I've discovered quite a few things that I at least had thought I had proved were impossible.

Of course, once I actually found out what's true, I could see that there were bugs in the proof.

But at least until we have things like Bruno's Theorema up and running there's no way at all to detect those bugs. It's not like *Mathematica*, where we can do all kinds of automated software quality assurance and so on.

We have to rely on human reviewers in journals and things like that, which is a very unreliable basis I think for finding out what might be true.

Well even though I myself don't happen to be particularly thrilled with it, mathematics as an activity has taken proof pretty seriously.

In fact, it's taken it so seriously that it hasn't really looked much at areas where it seems like you can't do proofs easily.

And essentially I think that's why there's never been mathematics that's looked for example at my friend the rule 30 cellular automaton.

In fact, I think it's a pretty good model for how mathematics has grown—that there's a pretty good model that one can make that's pretty much this.

It all started sometime in ancient Babylon from plain arithmetic and geometry. And those in turn actually arose from practical applications in commerce and land surveying and so on.

And in these areas, some neat theorems were found, and some proofs were developed for those theorems.

And being extremely pleased with the theorems, mathematicians started looking for ways to get the most out of these theorems.

And to do this, what they tended to do was to generalize the constructs that these theorems dealt with in such a way that the theorems always stayed true.

So the idea was to make things as general possible, making that generalization by having the constraint that some theorem or another could still be proved, but then sort of take away as many pieces as possible while still letting the theorem be proved.

So of course with this scheme one thing one can be sure of is that the systems that one was going to set up, were ones where the idea of proof still worked.

Well, one of the questions that's often asked is a question about whether mathematics is sort of invented or discovered.

I think it's pretty clear that the sets of rules that mathematics has actually ended up looking at were very much invented—in a definite human way, with particular constraints.

I suppose it's kind of weird: one usually thinks that mathematics is somehow more general and more abstract than, say, physics, or some other kind of natural science.

And the reason for that is that one thinks its rules are somehow more arbitrary.

It's somehow dealing with more arbitrary kinds of rules.

But actually the conclusion that I've come to is that that's not true, and that in fact the rules in physics—while there are many fewer of them—are chosen in a sense more arbitrarily—and are probably much more representative of all possible rules than the ones that have typically ended up being studied in mathematics.

So, in a sense, my adventures in looking at what arbitrary programs do can be thought of as a big generalization of ordinary mathematics—in a sense a study of what arbitrary mathematicses in the space of all possible mathematicses do.

Well as I'll talk about a little later, an arbitrary mathematics has some similarities to our particular human mathematics—the one that's developed historically—but it also has some pretty substantial differences.

But before I go on and talk about that, let me say a little about how all this relates to *Mathematica*, and to thinking about what one can call the foundations of *Mathematica*.

Well, first let's sort of say what my kind of abstract view of *Mathematica* is.

I've kind of viewed my job in designing *Mathematica* to be to think about all possible computations that one might want to do, and then to identify definite chunks of computational work that would get done over and over again in those computations.

And then what I've tried to do is give names to those chunks. And those are the primitive functions of *Mathematica*.

Well, one of the things I've learned from building my new kind of science is that I did a decent job in a certain way: the constructs that I put into *Mathematica* correspond pretty well with the constructs that seem to be relevant for representing for example the rules in our universe and in the natural world.

That has a very practical consequence. If you look at my new book when it's out, you'll find the notes in the back are full of *Mathematica* programs. And those programs implement the various kinds of programs that I talk about in the book.

But the point is that those implementations are mostly very very short. In other words, it doesn't take many *Mathematica* primitives to build the kinds of constructs one needs to represent the things that I think are going on in our universe.

Here's an extreme example of that: one of the things I've been doing a bit of is taking what I've discovered, and trying to use it to finally come up with a truly fundamental theory of physics. In a sense I've been trying to reduce physics to mathematics: to get a single abstract structure that is, exactly, everything in our universe.

That sounds like an extremely tall order, but for reasons I'm not going to go into here, I'm increasingly optimistic about my chances of success.

But the point is that one of the ways one can measure how well the primitives in *Mathematica* were chosen is how complicated the final program that is the universe needs to be, when it's written in *Mathematica*.

And actually, I had thought it was going to need to be rather long. But I noticed a year or so ago that in fact the basic system I was studying could actually be written as a surprisingly short collection of somewhat bizarre rules for *Mathematica* patterns. The only problem was that they didn't run all that fast.

I suppose that's not surprising if you're trying to reproduce the whole universe, that it not run all that fast.

But in the next version of *Mathematica*, or perhaps the one after that, I will tell you now the slight secret that there'll be a few hidden optimizations that make the universe run a little faster in *Mathematica*!

Well, I'm pretty confident that the primitives in *Mathematica* are well chosen if one wants to represent fairly arbitrary computations of the kind that for example seem to get used in natural science.

But what if one wants to "do mathematics"? Are the primitives that one has the best possible primitives?

Well, obviously one can go a very long way with these primitives, as we've all done.

And actually that right there tells one something about what's involved in mathematics.

But obviously there are a lot of primitives in *Mathematica*—say things like **Factor** and **Expand** and so on—that are very specifically set up to fit into particular kinds of structures that people often use in mathematics.

And then way down at the bottom level in *Mathematica* there are general transformation rules, which seem to be a good representation for arbitrary computations.

But the question is: what are the intermediate constructs that are not specific to particular areas of mathematics, but are constructs that support the general kinds of rules that are used in mathematics?

Well, to answer this one has to know more about what mathematics actually is, and what special characteristics its rules have.

One might think that the question of “what is mathematics?” is just one of these abstract things only of interest to philosophers.

But actually if one’s trying to design the most general stuff that will be useful in *Mathematica*, for mathematics, it becomes very important in practice.

And I should say immediately that I haven’t figured the answers here out, though I think I’m now beginning to define at least some of the relevant questions.

I kind of like these kinds of problems; in fact, in a sense this is ultimately the kind of thing I’ve spent most of my adult life doing: Trying to kind of whittle things down to find the very simplest most minimal constructs that one needs to do things. Whether that’s cellular automata and models of nature, or whether that’s constructs to support programming in *Mathematica*.

Of course it’s the fate of someone like me who spends a huge amount of time figuring out things like this that all these things in the end come to seem obvious.

Actually I’ve often thought how terrific it would be if people would study a little more carefully why things in *Mathematica*, for example, are done the way they are. There’s often a huge amount of thought behind things that might seem obvious, and having that thought more widely understood would be really good for everyone.

Anyway, I’m getting off-topic.

I was talking about understanding what the essential features of the activity that we call “mathematics” really are.

So how does one start on a question like this? Well, like any natural scientist I think the place to start is to build a model.

By the way, just in case there are any official mathematical logic model theorists in this audience, I don’t mean quite your kind of model. I mean the kind of model a physicist, for example, builds.

And the essence of that kind of model is that it’s somehow an idealization of a system.

What one tries to do in making a model in natural science is to capture those aspects of a system that one cares about, and ignore all the others. A model is in the end an abstract thing. I mean, one doesn’t think a planet going around its orbit has a bunch of little cogs—or little *Mathematicas*—inside it solving differential equations.

Instead, the differential equation is an abstract representation of the effect of what happens.

Now, I must say that scientists—even physicists—regularly get extremely confused about this issue.

I mean, the number of times I've heard physicists say: but how can your cellular automata model be a model for such-and-such, because we know the thing itself is solving a differential equation?

Well of course that's not true, both the differential equation and the cellular automaton are just abstract representations of an idealization of what the system is doing.

The bottom line is that a model, as that term is used in natural science, is an abstract idealization of something.

OK, so what's a good model for mathematics?

Well what we need is something that captures the essential features for our purposes of mathematics, but leaves all the inessential details out.

Well, I think it's been fairly clear for a century or so that the first step is to think about mathematics in terms of operations on some kind of symbolic structures.

And there are areas of mathematics like category theory that go a certain distance in defining general operations on symbolic structures.

But I've never thought they go nearly far enough.

And when I was designing *Mathematica* what I ended up trying to do was to set things up so one could go much further . . . so one could set up absolutely arbitrary symbolic structures and do essentially completely general transformations on them.

I originally came to this—actually when I was working on SMP around 1979—by thinking about how to capture and generalize what humans do when they're doing mathematics by hand.

I sort of imagined that one was always looking up books and finding rules that said that an expression like this gets transformed into an expression like that.

And then I kept on generalizing that idea, and ended up with the notion of transformation rules for patterns.

And, as we all know from *Mathematica* that idea is extremely successful in representing all kinds of computational and mathematical ideas. And gradually more and more people even seem to understand that point.

Well, what I did with transformation rules in SMP and later *Mathematica* is a bit like the whole tradition of work in mathematical logic and in the theory of computation.

But it's different—at least in intent—in some subtle but rather important ways.

And to understand more about what people usually think of as being the discipline of mathematics I'll have to explain something about that.

Ultimately the big difference is between being interested in doing proofs and being interested in doing calculations.

But to see what's gone on I'll have to tell you again a little bit about the history of all this.

Well, I guess it was at the end of the 1800s, particularly following all the discoveries made by generalizing Euclid's axioms and so on, there got to be the idea that somehow mathematics could be thought of as a study of the consequences of appropriately chosen sets of axioms.

And it turns out that axioms in this sense are very much like transformation rules for symbolic expressions.

These are Peano's axioms.

$$\begin{aligned} &\forall_x S(x) \neq 0 \\ &(\forall_x (\forall_y (S(x) == S(y) \Rightarrow x == y))) \\ &\forall_x x + 0 == x \\ &\forall_x (\forall_y x + S(y) == S(x + y)) \\ &\forall_x x \cdot 0 == 0 \\ &\forall_x (\forall_y x \cdot S(y) == y \cdot x + x) \\ &A(0) \Rightarrow ((\forall_x (A(x) \Rightarrow A(S(x)))) \Rightarrow (\forall_y A(y))) \end{aligned}$$

So this is just an example of an axiom system. These are Peano's axioms for arithmetic. And one can see that they're—I'm not going to go into this in great detail—one can think about these things as essentially like transformation rules for symbolic expressions.

Well actually, in the way things are often set up, there are usually so-called “rules of inference” in addition to axioms. And it's these rules of inference that are the direct analog of transformation rules.

But actually, as I'll perhaps explain later, one can just as well think about the axioms themselves as being like transformation rules.

And essentially what the axioms say is that one mathematical construct or statement can be transformed into another mathematical construct or statement.

OK. Well that sounds very much like transformation rules in *Mathematica*. But there's a crucial difference, and it's associated with the word “can”.

You see, in *Mathematica*, one just takes an expression and then uses a sequence of transformation rules and looks at what comes out.

But in the axiom setup in mathematics, one is usually interested in asking whether one can find any sequence of transformation rules that will get one from one particular expression to another.

It's basically the difference between calculation and proof.

In a calculation one just wants to follow a procedure and get an answer.

In a proof one wants to know whether there's any path one can follow that goes from one statement to another.

And as a practical matter it's usually much easier to do calculations than to do proofs.

And that's sort of why *Mathematica* can work well, and why logic programming languages like Prolog and so on that are closer to emulating proofs never really work that well.

And that's why I've always tried to be very careful in designing *Mathematica* to set modest expectations with the names of functions like **Simplify** that have to do things that are more like proofs.

But anyway, the idea of thinking of mathematics as the study of the consequences of axiom systems did introduce the notion of transformation rules historically.

But so what were these transformation rules like?

Well, actually, some of them were actually formulated very much like the ones in *Mathematica*.

And in fact by the 1940s there were transformation rules being talked about that are very similar in spirit to the ones in *Mathematica*.

But at least when people thought about these being applied to mathematics, they normally thought about them as being used in a proof kind of way rather than in the kind of way they are used in *Mathematica*.

OK, so basically what we've learned here is that the kinds of transformation rules that are in *Mathematica* are decent models for the kinds of transformation rules that are in mathematics.

And of course in a sense the success of *Mathematica* already told us that.

But one question is, do we need all the complicated stuff that's in the transformation rules in *Mathematica* to be able to reproduce what are somehow the essential features of mathematics?

Well, I don't think so. Not at all in fact.

And actually that's analogous to what I've found over and over again in physics and biology and other natural sciences. The really important and general phenomena don't depend on very much, so one can perfectly well capture them even with very simple models.

OK. So what then is the appropriate minimal model for mathematics?

What can we get rid of and still have the most important phenomena in mathematics?

Well, *Mathematica* patterns have already gotten rid of all sorts of hair that one usually sees in formulations of mathematics.

For example, there are no explicit types in *Mathematica* expressions. Only implicit ones from the structure of the expression tree or the names of heads.

I might say that when *Mathematica* was young, people often said: you can't do anything like mathematics without having an explicit notion of types.

But actually I think it's fairly clear that they were wrong. What we can do in *Mathematica* with symbolic expressions and their structures is actually much more general than what one can do with ordinary types.

Let me make a brief historical digression, that some of you may find fun, about types.

Well, in an effort to build up all of mathematics from some kind of uniform set of primitives the first really serious efforts along these lines got made by Frege in the late 1800s and by Russell and Whitehead in the early 1900s.

All of these folks had the idea that one should start from logic to try to build up all of mathematics.

For various reasons, I actually happen to think that the basic idea is fairly silly—that one wants to start actually from much more arbitrary symbolic structures, rather than from the particular structure defined by logic.

But anyway, what they were trying to do was a little like what I'm talking about today in making minimal models for mathematics—or, for that matter, what I was trying to do in the underlying design of *Mathematica*.

They wanted to be able to have a small set of primitives, and then assemble these to represent all the constructs of mathematics.

Well, they had a difficult time doing what they wanted to do. And of course they had many practical disadvantages compared to what we can do today. Like never being able to run the things they created.

But all in all I've actually always considered *Principia Mathematica* of Russell and Whitehead to be perhaps the worst example of language design of all time. I think some modern languages and systems are pretty bad. But actually nothing compared to *Principia Mathematica*.

One of the most baroque pieces of *Principia Mathematica* in fact has to do with types—and later ramified types—which were originally introduced as a way to avoid various logical paradoxes and so on.

Well, anyway—it's all a long story, not all of which I even know—but this idea of types that kind of came from this detail about avoiding paradoxes and this rather baroque formalism of *Principia Mathematica* somehow got assumed to be fundamental to all of the formalism of mathematics.

But I think *Mathematica* proves that it isn't really necessary to think about that kind of thing.

So, alright, let's assume *Mathematica* transformation rules are enough to represent mathematics. But what can we drop then in these transformation rules?

Well, another big feature of transformation rules is variables, and the ways variables are treated in scoping and things like that.

Another big feature of transformation rules, at least in *Mathematica*, is that they operate on expressions that can be thought of a bit like trees.

But let's for the time being assume we can just ignore both of these ideas.

Later, if there's time, I can talk a little bit about some rather abstract things called combinators that actually don't ignore these ideas, but still give very simple models of mathematics. But the models are a bit harder to understand than the models that I was planning to talk about.

OK, so we're going to ignore variables and tree structures.

What's then left in *Mathematica* and its transformation rules and so on?

Basically what *Mathematica* is then doing is transforming sequences of symbols, or strings of symbols.

Well, in a practical use of *Mathematica* the appropriate strings tend to be pretty complicated.

In an effort to find the minimal stuff that's going on, let's see what happens if we work only with very simple strings.

OK. So let me try and show you some examples of what an absolutely minimal *Mathematica* would look like.

Well, let's formulate this in *Mathematica*. Let's have an object **s** that I'm going to say is **Flat**.

I'm going to define various rules for **s** of various sequences of elements. And then all I'm going to do at every step is to apply these rules to the sequence of elements I've got, just following *Mathematica*'s normal how-it-applies-rules scheme.

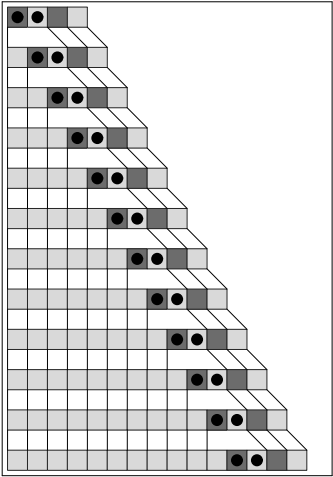
So let's take the rule $\mathbf{s}[1,0] \rightarrow \mathbf{s}[0,1,0]$. This says "1,0 gets turned into 0,1,0".

```
NestList[# /. {s[1, 0] -> s[0, 1, 0]} &, s[1, 0, 1, 0], 6]
```

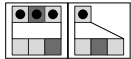
OK? Let's see what that does.

```
{s[1, 0, 1, 0], s[0, 1, 0, 1, 0], s[0, 0, 1, 0, 1, 0],  
 s[0, 0, 0, 1, 0, 1, 0], s[0, 0, 0, 0, 1, 0, 1, 0],  
 s[0, 0, 0, 0, 0, 1, 0, 1, 0], s[0, 0, 0, 0, 0, 0, 1, 0, 1, 0]}
```

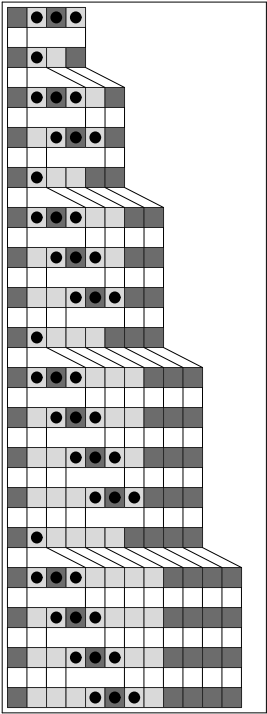
Well, we can make a picture that corresponds to the outcome that I got here, and it says that at every step that rule just prepends a "0" onto the string that we have. OK?



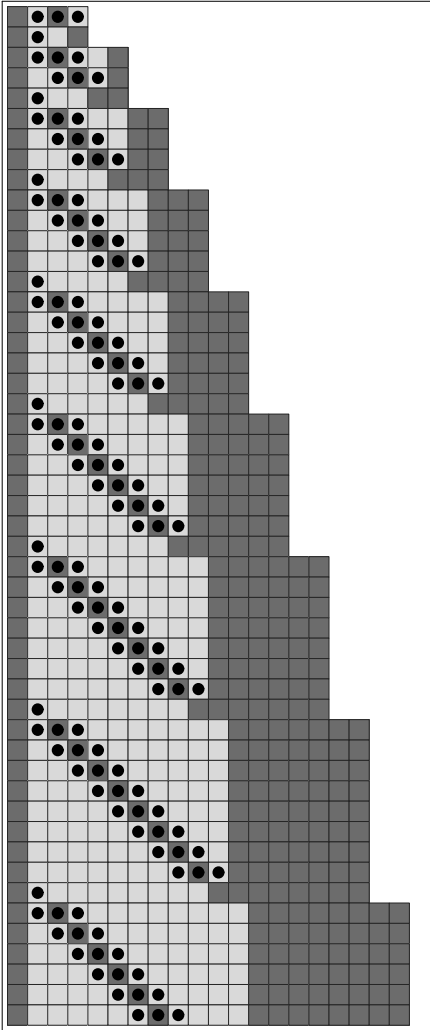
Well so that's an extremely trivial example of what *Mathematica* can do, so to speak. Let's try looking at another rule. Let's look at this rule here. This rule has two pieces in it. But it's, again, it's the same structure of rule.



Let's see what that does if we just start it off from some string. So here's what that does after a few steps.



So you can see it's doing something a little bit more complicated. And again what this is doing is it's just doing a sort of minimal version of what *Mathematica* does when it applies transformation rules. Here's a slightly bigger version of what happens with that rule.

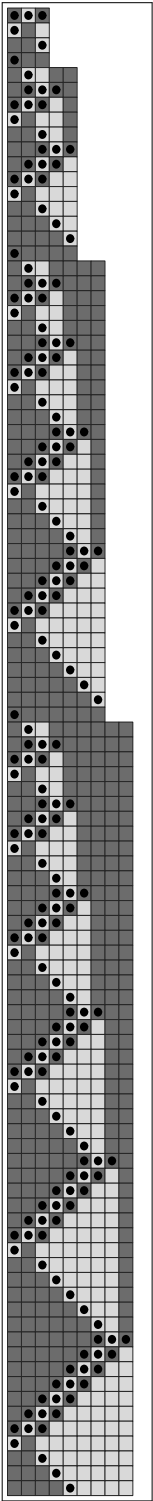


So again, just like our intuition might have suggested, if we have sufficiently simple rules here, we're doing the minimal version of what *Mathematica* does, the results are simple just like the rules are simple.

Let's try another example. Here's another rule.

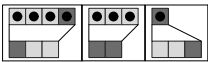


OK? It's slightly more complicated. Now there are three replacements in our list of *Mathematica* rules. Let's see what that one does.



A little bit more complicated. You can just see it going down the left. It’s a little bit more complicated. You can see it’s doing this sort of nested, kind of fractally thing. But again, it’s comparatively simple.

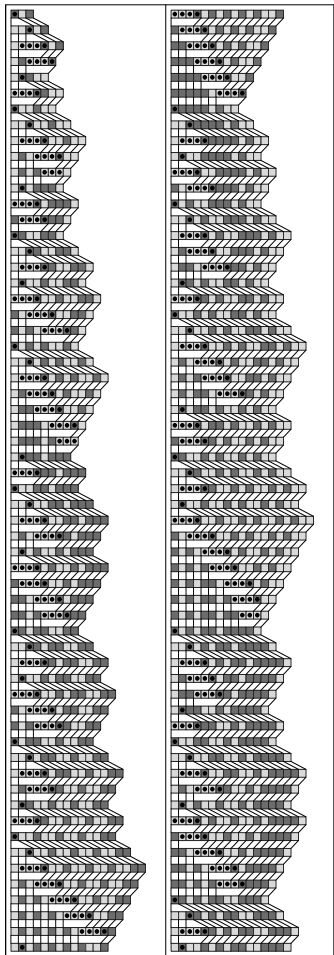
OK, let’s try another example. This is another set of sort of minimal *Mathematica* rules. Pretty simple set of rules.



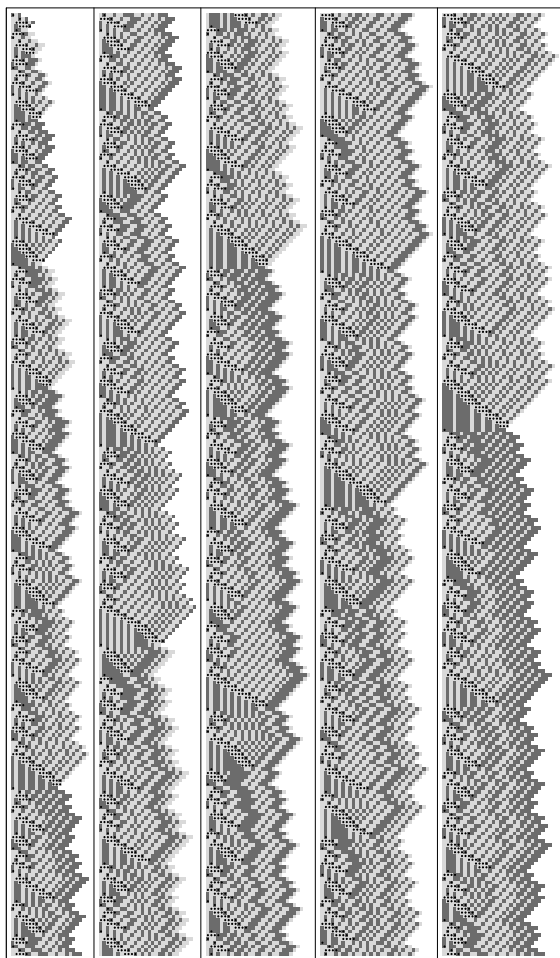
We could just write it down in *Mathematica* notation so to speak. In *Mathematica* notation, that rule would just be this thing here:

$$\{s[0, 0, 0, 1] \rightarrow s[1, 0, 0], s[0, 0, 0] \rightarrow s[1, 1], s[1] \rightarrow s[0, 0, 1]\}$$

Let’s see what that rule does if we apply it to some particular initial condition. Say we start off with “1 0 1”.



OK, that's what this rule does. Pretty simple rule as I showed, pretty simple initial condition. But that's the result it generates. Let's try running it for a bit longer. So what I've done here by the way is I've kind of folded this around so it goes down the first column then up to the top of the second column, and so on. Let's try running it, say just for a few hundred steps.



There's what we get. Again I've folded these columns around. So this is again an example, just like in my rule 30 cellular automaton, of a case where we have very simple rules, we start off with a very simple initial condition, yet the behavior we get is something very complicated.

And notice that this is a minimal idealization of *Mathematica*. This is not something like a cellular automaton that we got from somewhere else. This is something that just comes from essentially looking at an idealization of *Mathematica*, and in a sense an idealization of mathematics.

So what can we say about pictures like this.

Well, one pretty much immediate thing that one can say is that it can end up being undecidable what actually ends up happening in a picture like this.

I don't think I have time to explain lots about universal computation, and about a much stronger phenomenon that I call computational irreducibility.

But let's see what it might mean to say that something is undecidable in a system like this one.

Let me use a cellular automaton as an example. Let me pick a particular cellular automaton and let's start off running the cellular automaton with an initial condition that has just a single gray square in it. Well it's very easy to decide what will happen to the cellular automaton. After a little while the activity will halt, and the pattern will die out.



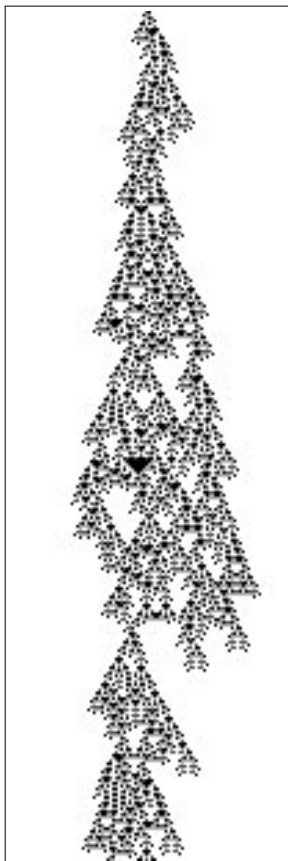
But let's say we change the initial condition



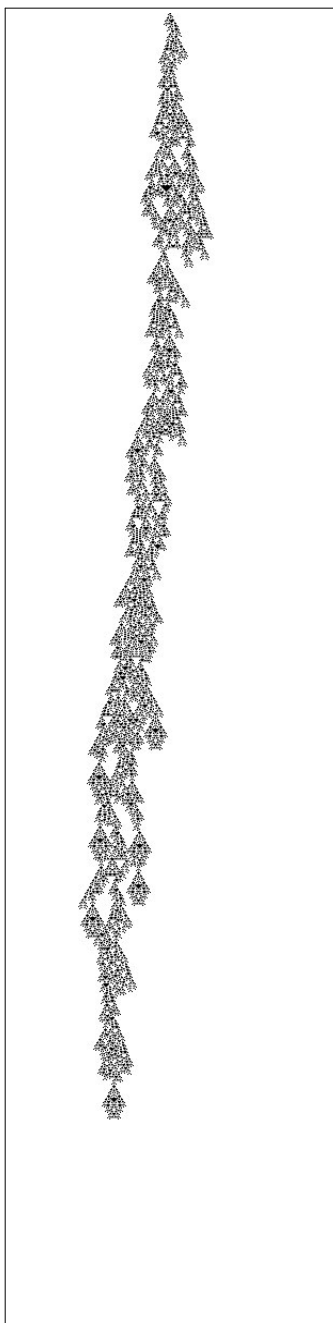
a little bit. Instead of having just a single "1" [light gray], we have a "2 3" [dark-gray cell, black cell] in the initial conditions.



So that's what it looks like then. It's a lot less clear what's going to happen then. Let's run it for 300 steps for example.



Is it going to die out? Is it not going to die out? How can we tell? Let's try running it for say 1200 steps. OK, and here's the result we get.

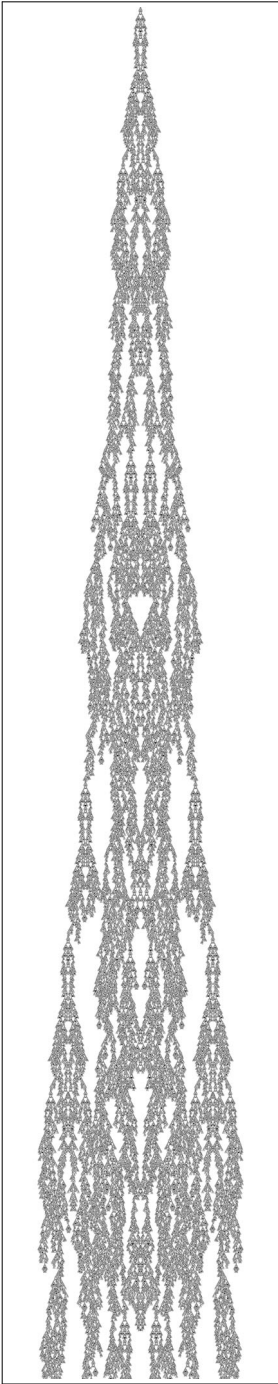


Here's the result we get from that. . . scroll down. . . well we see that sometime before 1200 steps, after doing all kinds of crazy things, this pattern eventually died out.

Well let's take another initial condition, let's say a pair of ones [pair of light-gray cells].



Let’s see what happens under those circumstances. Here’s what we get then.



And actually, I don't know what happens in this case. I can follow it for perhaps a million steps or so and it's still just kind of globulating around, and it sort of hasn't made up its mind about what it's going to do. It's not clear whether it's going to halt or not. And this is sort of a quintessential example of something where, the question of whether it ultimately halts or not, is something that will be formally undecidable. Well, how can one think about this?

If one thinks about trying to work out what will happen in a system like this, what one does when one tries to predict the system is one tries to have some kind of device that will outrun the actual evolution of the system. One can try to find some clever system that will be able to predict whatever the cellular automaton will do before the cellular automaton knows it.

But the best kind of predicting device will end up being a universal computer. It turns out that this cellular automaton itself is almost certainly a universal computer. For those of you who know about these things, that's actually a rather surprising and stunning fact. But I won't go into that right now.

Anyway that means that one will never be able to have a predicting device that will systematically outrun the actual cellular automaton.

So one will never be able to tell for sure what will happen after an infinite time, except in effect by spending an infinite time figuring it out, just like the cellular automaton does.

And that's why one says that the halting problem of telling whether a pattern will ever die out is undecidable.

Well OK. So this undecidability thing also happens in the string rewriting systems that I talked about before.

And it also happens in dealing with transformation rules in mathematics. But the point is that one doesn't need all the hair associated with actual ordinary mathematics to get this phenomenon. It already happens in our very simple idealization.

But back to the idealization.

This string rewriting thing that I've talked about is a decent idealization of *Mathematica* and of the calculational style of mathematics.

But what about of the proof style of mathematics?

Well, it turns out to be pretty easy to get an idealization of that too.

The whole thing is that in the string rewriting system that I did above, I always did things in sort of *Mathematica* style.

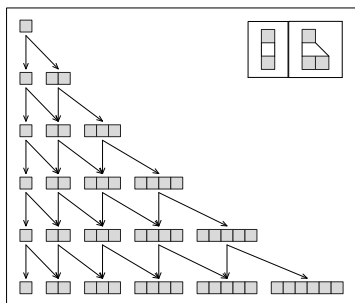
At each step, I just scanned the string, and applied the first rule that I could.

So at every step, there was a definite answer.

But in doing proof-style mathematics, one wants to do something different.

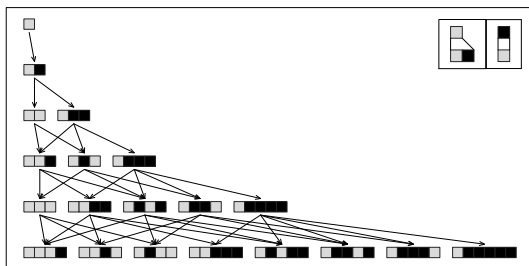
One wants to make what I call a multiway system, where one looks not just at a specific outcome from doing string rewriting in a particular way, but instead at all possible outcomes.

So let's see how that works. Here's an example of a multiway system. So the idea is that it's also a string rewriting system. This particular one uses these rules here—one square can get rewritten to one square, or one square can get rewritten to a pair of squares.



But instead of just rewriting the string, instead of just keeping the first rewriting that we end up doing, what we do is to look at all possible rewritings of the strings. So that means every little box here corresponds to one of the strings that can be generated, and there are arrows joining the boxes to show what can be derived from what by doing rewritings.

OK, so let me show you another example just to make clear what I'm doing here.



This one that I have here is a kind of Fibonacci-style string rewriting system, where white gets rewritten to white-black, and black gets rewritten to white. And so it's fairly easy to work out that there will be **GoldenRatio**ⁿ approximate—**Fibonacci[n]**—strings, at each of the 1, 1, 2, 3, 5, 8, etc. distinct strings at each step.

So what's the analog of a proof in this kind of system?

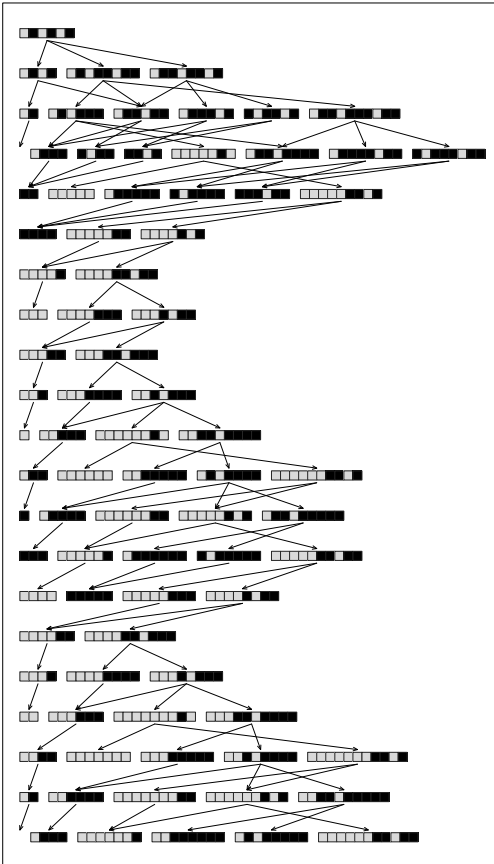
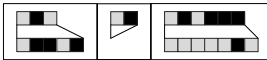
Well, it's actually pretty simple. A proof is like a path on this network. A proof is something that shows one that one particular string can be derived from another string. Remember that the axioms were these transformation rules here, which showed how strings can be rewritten. And the point is that what one is trying to do when one derives theorems is essentially to make up sort of super-transforma-

tion rules—things that for example say well if you have the string white-black-white, it can get rewritten to white-white-black-black-black down here.

And one could then add that as an axiom, to say that white-black-white can turn into this thing and then see what could be produced that way, and in the end the set of strings that one would generate would be the same as the set of strings that one generates just by following these axioms, but one would be able to do it quicker.

So essentially adding theorems, deriving theorems and adding theorems allows one to collapse the network, and get new theorems more easily.

Well, what can one tell from looking at these multiway systems? One question is, what do they typically do? These ones that I've shown you here do pretty simple things. But let's see what the typical multiway system does. One can sort of pick here essentially the axioms for the multiway systems at random and one can ask, what then is the resulting behavior of the network of theorems associated with this multiway system. So here's an example of a rule for a multiway system, and let's see what that one does.



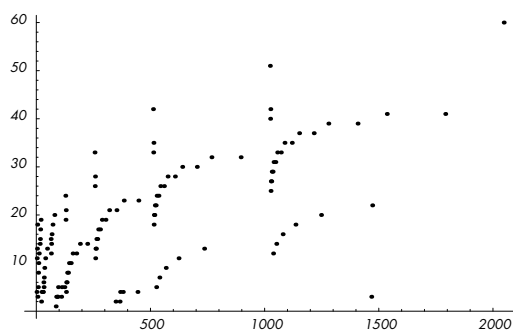
That one actually does something quite complicated. It's a pretty simple rule, yet again, like all of my things here. But what it does is quite complicated.

And, for example, I can ask many kinds of things about this system. For example, I could ask, starting off from this particular initial string, "when does the string that just consists of a single square first occur?" And the answer is it takes quite a few steps before that short string first occurs. Actually if you look at all possible multiway systems, most either don't generate many strings at all, or generate very rapidly increasing numbers of strings.

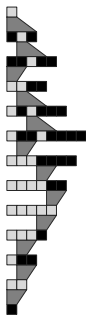
But it's not hard to find these weird and funky multiway systems that don't generate terribly many strings, but the number of strings they generate varies wildly with time. They don't have to have terribly complicated rules, and so it's fairly clear, from the fact that you can get multiway systems like this, that it ends up being essentially impossible to tell how long a chain of strings one will have to go through to get even from some fairly short string to some other short string.

And that's essentially directly analogous to the observation in mathematics that even pretty simple theorems can have very long proofs. But we're already seeing that phenomenon in our extremely simplified model of mathematics.

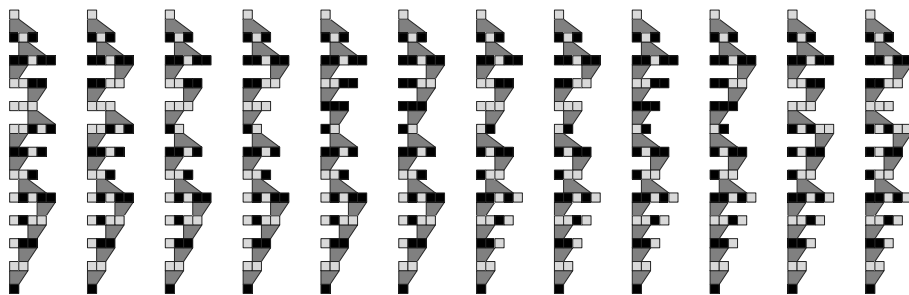
And because we've got a nice simple model of mathematics, we can ask questions about what the distribution of lengths of proofs for propositions of certain lengths is. For example, this is a picture that shows those strings which are eventually generated which correspond essentially to those propositions that turn out to be true, and that turn out to be provable. And it shows how far you have to go before you can generate those things.



I might say by the way, let me just show you another example of a slightly long proof in one of these systems. This is a written out proof in one of these systems. This says you rewrite a white square to black-white-black, you then rewrite the white square in the middle there to black-white-black, in this way, and you're kind of writing out a proof here, to show how you get from a white square to a black square.



And in this particular system there's actually more than one way to get from a white square to a black square, actually these are all the possible shortest proofs that correspond to getting from a white square to a black square.



OK. Let's go back to talking about phenomena in mathematics and how they relate to phenomena in this kind of simplified idealization of mathematics.

Well one very famous phenomenon in mathematics is Gödel's Theorem. And I thought I might tell you a little bit about how Gödel's Theorem is related to what I've been saying.

Well I have to admit something about the models I've been using so far. They are a bit different from what people consider usually as mathematics, because they don't have logic, they don't have a notion of logic explicitly built into them. As I mentioned before, most of traditional and proof-based mathematics is ultimately based on logic.

I'm not sure that's the best thing, but that's the way it is.

The kinds of multiway systems that I've been showing you so far, are fine ways of representing relations in mathematics—but not quite theorems.

Actually some people may recognize my multiway systems as being things that are sometimes called semi-Thue systems.

And these are essentially semi-semigroups. You see, you can think of the transformations that I'm doing here as being like relations between words in an algebraic structure. I'm going to get slightly technical here but it might be of interest to some people and help people to understand what's going on.

In my systems the transformations can be absolutely anything. But if every transformation can be applied in either direction then it corresponds to the relations of a semigroup. And if in addition there are inverse elements allowed, one has a group. Now, being even more technical, I should say that the pictures I'm making you might think were Cayley diagrams, they're not. They're actually a kind of lower level construct. They're pictures of sort of what's inside each node of a Cayley diagram—of all the words that are equivalent, say, to the identity.

OK. Well, some of you may know about the undecidability of the word problem for groups. That's essentially the same statement I'm making—about the paths that go from one word to another by applying transformations being arbitrarily long.

Let me talk about a more familiar example. Let's think of the poor **Simplify** function in *Mathematica*. It works by applying various transformation rules to an expression. Let's say that we're trying to determine whether some particular expression is zero. Well, some transformations may make the expression smaller, but some may make it bigger. And we can end up having to go on an arbitrarily long path in the attempt to find out whether our expression is zero or not. And actually that's the same phenomenon I've shown you too.

OK. But back to the questions about theorems. I've said that these multiway systems of mine don't really have logic built into them. The underlying problem is that they don't have explicit notions of True and False. They're just dealing with expressions, and transformations between expressions. If those expressions are supposed to be propositions—candidate theorems in some kind of mathematics—they should presumably be either True or False.

So how do we set this up? Well, it's really very easy. We just need some operation on strings that's like negation, and that turns a True string into a False one and vice versa. And so as an example the operation might just reverse the color of each element.

OK. So now how do we find all the True theorems? Well, we just start from the string that represents True, let's say a single black cell, and then we find all the strings that we can derive from it. But. . . there are some nasty things that can go wrong. If our rules—our axioms—aren't sensible, we might be able to derive two strings that are related by negation. And that would mean that we have some kind of inconsistency. There are two theorems we're saying are both true, but they are negations of each other.

And most mathematicians would then say that the axiom system one's picked is no good. But let's say we avoid this problem, and that we are sure nothing inconsistent like that happens. Then are all axiom systems with this property decent axiom systems for mathematics?

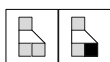
Well, there's another problem. And that problem is completeness. Given a theorem, can we actually determine from the axioms whether the theorem is True or False? Well, in terms of our networks that's an easy question to formulate.

We want to know whether anywhere on the network our particular proposition, our particular theorem will appear, starting from the string that corresponds to True. Well, it's pretty obvious there are lots of axiom systems where tons of strings will never appear—axiom systems that are utterly incomplete. But what would a mathematician think of one of those axiom systems? They'd probably think for many purposes that they were kind of stupid.

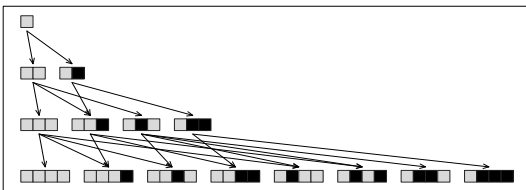
Let's say one had an axiom system that was supposed to have something to do with arithmetic. And it was all very nice. But it couldn't prove that addition is commutative. Mathematicians would probably say that that was a dumb axiom system; that they need a more powerful axiom system.

OK, so here's the issue: is there any axiom system that's always powerful enough to be complete? Well, it's actually quite easy to find one.

Here is an example of an axiom system that is complete and the list of strings that it generates. The axiom system has two rules in it, like this,



it starts off from the string that represents True. And what one finds is that it generates exactly half of the possible strings of a given length. Let's see, I have to figure out what the interpretation of this thing is—but anyway, it generates exactly half the strings of a given length, and you can apply negation to each of these strings and find out that it doesn't generate the string that is the negation of that string.



OK. This axiom system is both consistent and complete, but it's fairly trivial. Well how about something that has all the richness of a real area of mathematics?

Well, Hilbert showed that Euclidean geometry is an example: something that is complete and consistent. And people thought in the early part of this century, that all other serious mathematical axiom systems would be the same way.

But what happened in 1931 was that Gödel showed that that wasn't true. He showed that Peano arithmetic was incomplete—that there were propositions in Peano arithmetic that couldn't be proved in any finite way.

Again, one might say, so what? I mean, in what's called Robinson arithmetic one has a fine set of axioms, but it so happens that it's fairly obvious that the system isn't complete. Because, for example, you can't prove that addition is commutative. So why is it a big deal that Peano arithmetic isn't complete?

Well, it turns out that it's really a quantitative issue, not a qualitative one. The point is that Peano arithmetic is very very nearly complete. I mean, all the simple propositions are connected—or their negations are connected—to the big network of all True propositions of arithmetic. But what Gödel showed is that there are some propositions that aren't connected to this big network of true propositions of arithmetic. It's a network that might look something like this, except that it grows much more rapidly at every step.

It's a bit of a long story, but basically Gödel used the idea of universal computation to compile the statement "this statement is unprovable" into the very low-level machine code of Peano arithmetic, and that's how he got his result. But the machine code of that particular expression—the proposition in sort of algebraic terms that's equivalent to "this statement is unprovable"—is incredibly long. So that's what was so surprising about Gödel's Theorem. That Peano arithmetic could seem almost complete—but not actually be complete.

Well, one might ask whether Gödel's statement is the shortest statement that isn't provable in Peano arithmetic. And it's been known for some time that it's not. There are some slightly simpler ones. But they're still incredibly complicated. They're nothing like Fermat's Last Theorem, or some kind of question that mathematicians might seriously ask.

Well, as probably most of you know, Gödel's Theorem was a big deal when it came out, because it showed that Hilbert's ideas about mathematics, and about being able systematically to establish everything from axioms, weren't going to work. But as a practical matter, in the past 68 or so years, nobody in practical mathematics has ended up paying all that much attention to Gödel's Theorem. It's always seemed too remote from the kind of questions that mathematicians actually end up asking. And the reason for this is that in Peano arithmetic the simplest incomplete propositions—that are known at least—are really complicated.

Well, I happen to think there are some somewhat simpler ones—I even have a potential candidate for one. But still, Peano arithmetic is basically a special almost-incompleteness-safe axiom system.

So... what does this mean about our efforts to find simple models of mathematics? Is this feature of Peano arithmetic something that's very hard to get? And that requires all the hair of Peano arithmetic and that's important in having something that is a realistic model of what mathematics could be.

One might think so. But it's actually not true. Let me just say that if one searches through simple multiway systems, one can find other examples. Some systems are obviously inconsistent; some are obviously incomplete.

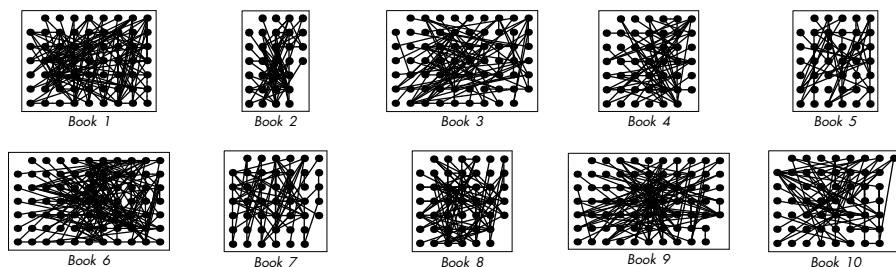
But it's not hard to find ones where incompleteness happens sort of arbitrarily far away. So there's nothing special about Peano arithmetic in this respect either.

So, OK. What are all these axiom systems that seem to have all the familiar math properties? Well, you've never heard of any of them. They're just not things that have ever been reached in the history of human mathematics. And there's a huge world of them out there. Some have easy interpretations to us humans. Some don't. They're sort of alternative mathematicses. So how does one compare these mathematicses to the mathematicses we're familiar with?

I mean, each field of mathematics has a different character. Even within the mathematics that we are familiar with, different fields have different characters. How does that show up in the idealization of mathematics that I've been talking about?

One can think of building this network of theorems for each field. Say for number theory. Or geometry. Or functional analysis, or something like that. And then one could for example start asking simple questions like, what the statistical properties are. How connected it is? How many long paths there are in it?

Let me show you for instance for Euclid, Euclid's Elements. This is the dependency graph of the propositions of Euclid's Elements, in Book 1, Book 2, Book 3, etc.



You can unravel these to some extent. I haven't got the best unraveling yet. Euclid was a little bit optimistic in what needed to have what depend on. But anyway, this is the network that you get from looking at the dependencies in Euclid's Elements.

I was showing you this rather messy dependency network. This is the picture of the particular theorems Euclid thought were interesting that you get from the multiway system that is geometry. It reminds me of a quote from Gödel, when asked about mathematics, what the role of a mathematician was, his idea was that the role of the mathematician was to work out what theorems are interesting. That a machine can generate all the possible theorems, but the role of a mathematician is to work out what theorems should be considered interesting.

Anyway, what do the properties of this kind of network mean in mathematics? When people do mathematics they talk about deep theorems, powerful theorems, easy lemmas, elegant theorems those kinds of things. And it turns out that each of those kinds of things, one can start to try to define in a precise way in terms of the properties of a network like this.

And I think if one drew the networks for, say, number theory and say, algebra, one would be able to see by looking at them something about the very different characters of those fields.

So one question one might ask is, “What common features are there in the networks humans happen to have thought up for their mathematics?” I don’t exactly know right now. That’s something for which one would require quite a bit of empirical data to be able to work out. But the critical question is whether it’s all just historical, or whether some of it is a consequence of some general principle—or perhaps some human limitation or human foible.

It’s a little bit like asking about human languages, whether there are all possible forms that appear in human languages or not. The actual observation is that while there are incredibly many exceptions, typical human languages that people actually understand are not that far from being context free.

Well, it’s also interesting to think about what kind of mathematics . . . and thinking about whether the mathematics that we have is necessary, to think about what kind of mathematics the extraterrestrials might have. People tend to assume that they’ll think primes are important.

I think that’s a fairly absurd possibility. I’d bet on the rule 30 cellular automaton over the primes any day. And actually I think from my studies of all possible simple programs, I can say a certain amount about what’s likely to certainly exist in the mathematics of any creature. But it’s definitely not the primes, I think. I mean, even in human mathematical history, there are lots of obvious twists and turns. If you talked to Pythagoras, for instance, he’d be much more excited about perfect numbers than about prime numbers. But it so happens that primes have won out in the history of mathematics in the way it’s developed. But I don’t think that’s in the slightest bit fundamental.

But OK. If we think about building *Mathematica*, *Mathematica* does reflect actual history. I mean, we have all those special functions that happen to have been identified in the nineteenth century, and so on.

And they’re very useful. There’s nothing wrong with us pandering to the history of mathematics in designing *Mathematica*.

It’s just that it would be nice to find constructs that are as general as possible to represent what happens in humanly chosen mathematicses.

So that’s a challenge. Now, if one tries for example to do the proof thing in complete generality, and thinks that the idea of proof in general is the thing that is special about human mathematics, then one will quickly fail.

I mean, let’s say we try to make a function—call it say, **FindRules**—that takes a list of replacement rules, and instead of just having a single expression that it tries to reduce—it has two expressions, and tries to find a sequence of replacements from the list that it’s been given that will take one expression into the other.

Well, this function will be very unsatisfactory. It’ll have to work by doing the same kind of thing that we have to do in one of these multiway systems. And it

will have a very hard time trying to find whether there happens to be a path from one of the expressions that we gave to the other expression that we gave.

We'd be getting tech support calls about this function all the time. And the reviewers would be doing benchmarks and complaining that the function is too slow, and so on. Because—and this is what I've discovered from looking at random simple programs—much of the time the network of things one can get to from the replacements is a big mess, and there's no quick way to find out what's going on.

In a sense, people would be reporting the undecidability of the halting problem as a bug all the time. But the question is: can one do better?

And there are two ways we might be able to do better. One is by using essentially a directly historical approach, and basing everything on the kinds of rules that happen to have shown up in arithmetic and geometry and all the things that have been derived from them.

But maybe, just maybe, one can do something more general. And to see if one can one has to understand what it really takes to be a human mathematics. I mean, if one looks at arbitrary mathematicses that come out from the simple rules and simple programs I've investigated, one is usually thrown into undecidability and so on very quickly. And in fact the whole idea of doing proofs and things pretty much disintegrates in that case.

But perhaps that isn't true for any set of rules that would come from axioms that might realistically be used in human mathematics. It's sort of the following question: if you see a random collection of axioms, can you tell if they had a human origin?

It's again a little bit like asking about languages. If one has an approximate formal grammar for a language—a natural language or a computer language—can one tell if it was set up or used by humans? Or whether it was just randomly chosen?

I'm not sure about the answer to this for mathematics, and it'll be interesting to see. One thing I am sure about is that if one doesn't put on whatever constraints are associated with human mathematics, there's an awful lot of other mathematicses out there to investigate. Mostly they can't be investigated by the same methods as the ones that have been used so far in history in human mathematics. Mostly I think the proof idea for example won't work. And instead one's left with—just as in most fields of science—is mostly finding out what's true by doing experiments, not by making proofs about what has to be true. And of course in doing those experiments that's something that *Mathematica* can do very well in its present form.

But even though there's a lot of other mathematicses out there that one might investigate—with all sorts of weird and wonderful properties—that doesn't mean there's no more to do with our regular old human mathematics. There certainly is. And I hope it can be done very well with *Mathematica*.

But as one tries to see how to do it, one has to understand what one is doing, and the fact that the mathematics that one is investigating is I think very much a historical, cultural, artifact. It happens to be a very wonderful artifact. Intellectu-

ally I think it's much deeper than for example systems of human laws, or our schemes for art or architecture and things like that. But nevertheless an artifact.

And that means that one cannot somehow expect it to be typical of all the possible mathematicses, or of what might happen in nature.

But that's what my new kind of science is trying to deal with: working out what all those possibilities are, what all possible mathematicses might do. And that's what I've spent the past eight years or so investigating, and in finding out how the mathematicses that are sampled by the natural world—what those are like and what their properties typically are.

Well, I should probably stop here. As is usual I've probably gone way over time. And I'm afraid this has gotten very abstract, but perhaps I've managed to communicate at least a little bit about a few of the things I've been thinking about. So I thank you all very much.

Maxillo-Facial Frame: Finite Element Shapes

Keynote address delivered by Jacques Treil

Gautam Dasgupta

*Professor, Civil Engineering and Engineering Mechanics
Columbia University
600 Seeley W. Mudd
New York, NY 10027, USA
gd18@columbia.edu*

Jacques Treil

*Docteur, Service de radiologie
Clinique Pasteur
45 Avenue de Lombez
Toulouse 31300, France
jtreil@cadrus.fr*

■ **Introduction**

A computational scheme has been developed to describe and analyze the human face. The study of an orthomorphic sampling has revealed an architectural balance of the anatomical components. The maxillo-facial model by Treil has now become a standard reference for scientists and medical practitioners.

The c2000 cepha software mathematically deduces the location of anatomical landmarks from CT scan data. Subsequently, three-dimensional images of the maxillo-facial anatomical structures are generated for human faces. The principal coordinate directions are determined on the basis of selected anatomical locations and teeth positions. All of these landmarks are situated on the trigeminal axes described by Moss [1], and they form an invariant frame of reference in the study of the maxillo-facial growth. Furthermore, via the calculation of these

Originally presented at the Third International *Mathematica* Symposium (Hagenberg, Austria, August 23–25, 1999).

principal axes, the model is organized into a hierarchy of anatomical elements—for example, the teeth, half-arches, maxillar and mandibular arches separately, both arches together, all elements as a group, and the maxillo-facial frame. Each element's position and axes of inertia define a three-dimensional reference landmark. This allows for the calculation of relative spatial orientations of anatomical elements. Biologically significant components of the growth tensor with respect to invariant frames of reference (determined on the basis of principal moments of inertia) are calculated by *Mathematica* packages.

Individual research pursued by the authors motivated innovative computing in medical and engineering sciences, with symbolic computations aided by graphics and sound. In this presentation, the following items will be developed using *Mathematica*:

- Consistency of the geometrical model for lower-order interpolants
- Morphometric considerations in bioengineering computations
- Formulation of finite/boundary elements with specific emphasis on radiology and three-dimensional reconstruction

In the oral presentation, collaborating efforts in imaging and mathematical representation of anatomical elements were emphasized. *Mathematica* constructs for handling CT data were elaborated; especially the deeper biological implications were derived from the formulation. Calculation of true (not projected) parameters and application of symbolic computer mathematics in radiology were illustrated using the *Mathematica* graphics environment. Anatomical concepts via tensorial invariants are related to biological objects. Consequently, the axes of a single tooth, the coronoradicular axis, the bucalingual axis and the mesio-distal axis were identified. Statistical calculations of the principal axes of inertia were presented using color graphics. *Mathematica* definitions of geometrical location and local and global axes of symmetry were employed to model ontogenic and philogenic growth.

Representation of the growth tensor according to the computer mathematics formulation confirms the Function Theory of Biology proposed by Moss [1]. In the future, Maeder's parallel computing strategy [2] will be implemented to transform symbolic objects of biological significance, and the resulting (very large volume of) data (120GB) will be handled in external C programs (such as MathCode [3]).

NOTE: Some inputs in this article depend on an extensive set of programs contained in several large notebooks. In the interest of brevity, we have omitted these programs, and the affected inputs appear in comments. Interested readers are encouraged to contact Gautam Dasgupta (gd18@columbia.edu).

■ Views of Treil's Frames

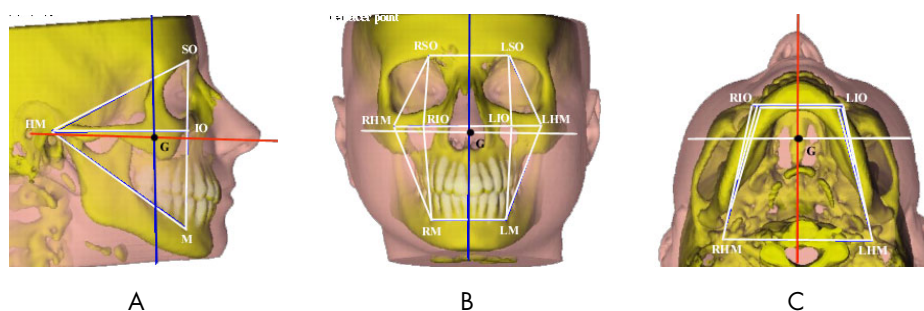


Figure 1. Here are three Sagittal A, Frontal B, and Axial C views of the maxillo-facial frame. The maxillo-facial frame is defined by eight anatomical landmarks: both supra orbital foramen (RSO, LSO), both infra orbital foramen (RIO, LIO), both mental foramen (RM, LM), and both head malleus (RHM, LHM). According to Moss theory, all these landmarks are located on the trigeminal neuro-matrical facial growth directions. The axes of inertia of the maxillo-facial frame are in blue the principal, in white the second longest, and in red the shortest.

This *Mathematica* notebook contains only the numerical issues covered in Treil's keynote lecture. The biological and clinical details can be found in the enclosed references.

The shape function interpolants are the basic objects in any finite element method. In morphometric analysis (i.e., quantifying shape and size changes), a kinematic formulation is adequate without any reference to stress or stress-strain-like constitutive relations. Hence the algebraic expressions of shape functions can be used directly to calculate the non-mechanical growth strains—the local changes in shape and size of the body—in time. In a finite element region with n -number of nodes there are n shape functions. Since the intent of the shape function interpolants is to reproduce m pre-selected algebraic shapes, it is shown here that from $n - m$ given shape functions the remaining m interpolants can be obtained. This new result substantially economizes the computing effort required to deduce the complete set of interpolants.

■ Treil's Frame Schematic

Treil's maxillo-facial frame [4, 5], according to the functional matrix hypothesis [1] is presented below.

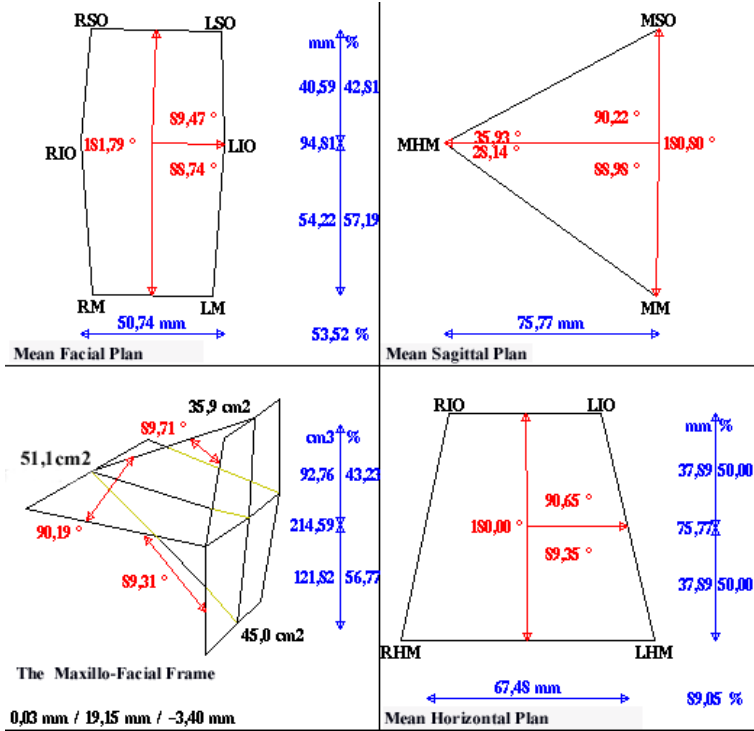
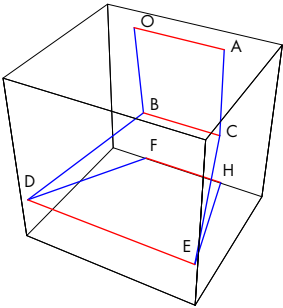


Figure 2. Treil's frame schematic.

The four projected views are shown above.

A symmetric wire-line model of the maxillo-facial frame is shown below.



The anatomical symbols: O, A, B, C, D, E, F, H in the maxillo-facial frame are represented by nodes with indices 1 through 8. The coordinates of the i th node is denoted by $\{x[i], y[i], z[i]\}$.

■ Maxillo-Facial Frames

There are eight nodes in the Treil maxillo-facial frame [4, 5]. Only one shape function will be evaluated in the $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ coordinates system following a suitable method, such as Taig's isoparametric formulation [6] or Wachspress' rational polynomial interpolation [7, 8]. (*Mathematica* coding of these two popular and effective procedures are included in the appendix.) This initial shape function in terms of $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ will be indicated by the special symbol $\mathbf{s}[0]$, and the associated coordinates will be denoted by $(\mathbf{x}0, \mathbf{y}0, \mathbf{z}0)$. Here the string "0" reflects the starting choice for the proposed algorithm. For the eight-node maxillo-facial frame the remaining seven interpolants, $\mathbf{Array}[\mathbf{s}, 7]$, will be calculated to reproduce the set of exact fields denoted by $\mathbf{f}$ as a list of functions in $(\mathbf{x}, \mathbf{y}, \mathbf{z})$. In particular, here: $\mathbf{f} = \{1, \mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{x}*\mathbf{y}, \mathbf{y}*\mathbf{z}, \mathbf{z}*\mathbf{x}\}$. Its algebraic relations are crucial to this presentation, as explained below.

Consider a pre-assigned field $\mathbf{u}[\mathbf{x}, \mathbf{y}, \mathbf{z}]$ to be an element of the list $\mathbf{f}$. With the prescribed nodal values $\mathbf{u}[\mathbf{x}[i], \mathbf{y}[i], \mathbf{z}[i]]$, the set of shape functions $\mathbf{s}[i]$ in terms of $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ should lead to

```
In[1]:= u[x, y, z] == Sum[u[x[i], y[i], z[i]] * s[i], {i, n}];
```

The aforementioned relation will be exactly satisfied in terms of the coordinate variables $(\mathbf{x}, \mathbf{y}, \mathbf{z})$. Hence the name "exact interpolated fields" for $\mathbf{f}$.

□ Mathematica Example

```
In[2]:= Clear[f, x, y, z, s];
        f = {1, x, y, z, x*y, y*z, z*x};
```

The following code generates the coefficient matrix **mat** and the right-hand side **vec** for **LinearSolve**:

```
In[4]:= shapeRelation // Clear;

shapeRelation::usage =
  "shapeRelation[f,s[0],{x,y,z},{x0,y0,z0}] generates {mat,
    vec} for determining the shape functions as LinearSolve[
    mat, vec] with the initial shape function s[0] associated
    with the node at {x0, y0, z0} by exactly interpolating
    the elements of f in the coordinate system (x,y,z).";

shapeRelation[f_List, g_, {x_, y_, z_}, {x0_, y0_, z0_}] := Module[{i},
  {-Table[f /. {x -> x[i], y -> y[i], z -> z[i]}, {i, Length[f]}] // Thread,
   -f + (g*f /. {x -> x0, y -> y0, z -> z0})}]
```

Execution

```
In[7]:= Clear[mat, vec];
        {mat, vec} = shapeRelation[f, s[0], {x, y, z}, {x0, y0, z0}];
```

A Symmetric Case

The anatomical points are symmetrically located on the left and right side of the face. The canonical maxillo-facial frame is then symmetric about x -axis. Hence the following is a specialization for which a symmetric frame is generated according to the rule.

```
In[8]:= Clear[r];
r := {x[1] → -x0, y[1] → y0, y0 → 0,
      z[1] → z0, z0 → 0,
      x[2] → -x[3], y[3] → y[2], y[2] → 0, z[3] → z[2], x[4] → -x[5],
      y[4] → y[5], z[4] → z[5], x[6] → -x[7], y[6] → y[7], z[6] → z[7]};
```

Now the coefficient matrix and the right-hand side vector, **matr**, **vecr**, respectively, become:

```
In[10]:= Clear[matr, vecr];
{matr, vecr} = {mat, vec} /. r;
```

```
In[12]:= MatrixForm[matr]
```

```
Out[12]//MatrixForm=
```

$$\begin{pmatrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ x0 & x[3] & -x[3] & x[5] & -x[5] & x[7] & -x[7] \\ -y0 & 0 & -y[2] & -y[5] & -y[5] & -y[7] & -y[7] \\ -z0 & -z[2] & -z[2] & -z[5] & -z[5] & -z[7] & -z[7] \\ x0 y0 & 0 & -x[3] y[2] & x[5] y[5] & -x[5] y[5] & x[7] y[7] & -x[7] y[7] \\ -y0 z0 & 0 & -y[2] z[2] & -y[5] z[5] & -y[5] z[5] & -y[7] z[7] & -y[7] z[7] \\ x0 z0 & x[3] z[2] & -x[3] z[2] & x[5] z[5] & -x[5] z[5] & x[7] z[7] & -x[7] z[7] \end{pmatrix}$$

```
In[13]:= vecr
```

```
Out[13]= {-1 + s[0], -x + x0 s[0], -y, -z, -x y, -y z, -x z}
```

Before attempting to solve for the unknown shape functions, the consistency of the assumption of the interpolated field should be established.

```
In[14]:= Simplify[Det[matr]]
```

```
Out[14]= 4 x[5] x[7] (x0 y[2] (y[5] - y[7]) (z[2] - z[5])
(z[2] - z[7]) (z0 (-y[5] + y[7]) - y[7] z[5] + y[5] z[7]) -
y0 (-x0 y[2] (y[5] - y[7]) (z[2] - z[5]) (z[2] - z[7]) (z[5] - z[7]) +
2 z0 x[3] (y[7] (-z[2] + z[5]) + y[5] (z[2] - z[7]))
(y[7] (-z[2] + z[5]) + y[5] (z[2] - z[7]) + y[2] (-z[5] + z[7])) +
x[3] (y[2] (z[5] - z[7]) (y[5] (z[2] + z[5]) (z[2] - z[7]) - y[7] (z[2] - z[5])
(z[2] + z[7])) - 2 (y[5]^2 z[5] (z[2] - z[7])^2 + y[7]^2 (z[2] - z[5])^2
z[7] - y[5] y[7] (z[2] - z[5]) (z[2] - z[7]) (z[5] + z[7])))) +
x[3] (-2 y[5] y[7] z[2] (y[7] (-z[2] + z[5]) + y[5] (z[2] - z[7])) (z[5] - z[7]) +
y[2] (-y[7]^2 (z[2] - z[5]) z[5] (z[2] - z[7]) -
y[5]^2 (z[2] - z[5]) (z[2] - z[7]) z[7] + y[5] y[7] (z[2]^2 (z[5] + z[7]) +
z[5] z[7] (z[5] + z[7]) + z[2] (z[5]^2 - 6 z[5] z[7] + z[7]^2)))) +
z0 (2 y[5] y[7] (y[7] (-z[2] + z[5]) + y[5] (z[2] - z[7])) (z[5] - z[7]) +
y[2] (y[5]^2 (z[2] - z[5]) (z[2] - z[7]) + y[7]^2 (z[2] - z[5]) (z[2] - z[7]) -
2 y[5] y[7] (z[2]^2 + z[5]^2 - z[5] z[7] + z[7]^2 - z[2] (z[5] + z[7]))))))
```

In general, the nonzero determinant signifies that the assumption:

```
In[15]:= f
```

```
Out[15]:= {1, x, y, z, x y, y z, x z}
```

could be acceptable and the matrix **matr** is indeed invertible.

For given numerical data for the nodal coordinates, the shape functions will be obtained from the expression below, after removing the comments (* and *).

```
In[16]:= (*
          Evaluate[Array[ss,7]] = LinearSolve[matr, vecr];
          *)
```

□ Failure to Interpolate: x^2 , y^2 , z^2 —An Example of Inconsistent Data

```
In[17]:= Clear[f, x, y, z, s];
```

```
In[18]:= f = {1, x, y, z, x^2, y^2, z^2};
```

```
In[19]:= {mat, vec} = shapeRelation[f, s[0], {x, y, z}, {x0, y0, z0}];
```

The Symmetric Maxillo-Facial Frame

```
In[20]:= (mat /. r) // MatrixForm
```

```
Out[20]//MatrixForm=
```

$$\begin{pmatrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ x0 & x[3] & -x[3] & x[5] & -x[5] & x[7] & -x[7] \\ 0 & 0 & 0 & -y[5] & -y[5] & -y[7] & -y[7] \\ 0 & -z[2] & -z[2] & -z[5] & -z[5] & -z[7] & -z[7] \\ -x0^2 & -x[3]^2 & -x[3]^2 & -x[5]^2 & -x[5]^2 & -x[7]^2 & -x[7]^2 \\ 0 & 0 & 0 & -y[5]^2 & -y[5]^2 & -y[7]^2 & -y[7]^2 \\ 0 & -z[2]^2 & -z[2]^2 & -z[5]^2 & -z[5]^2 & -z[7]^2 & -z[7]^2 \end{pmatrix}$$

```
In[21]:= Det[(mat /. r)]
```

```
Out[21]:= 0
```

Hence the following evaluation is not attempted since the coefficient matrix is not invertible.

```
In[22]:= (* LinearSolve[mat, vec] *)
```

■ Some 2D Results—Concave Quadrilateral Element

Standard literature in finite elements hardly addresses the convexity issue. There is no mention of it in the encyclopedic references in the two volumes of Zienkiewicz and Taylor [9]. Wachspress addresses the condition from a projective geometry point of view. He develops concavity with curved sides. His general procedure [10] for n -gon is not applicable since the curve through the external intersection points intersects the element domain. It is then prudent to conclude that there has not been any consistent method available to generate the shape functions of a concave polygon. Recently, Dasgupta utilized the two-dimensional

version of the aforementioned code in the form **shapeRelation[f,s,{x,y},{x0,y0}]** to generate the second, third, and the fourth shape functions from the **s** which pertained to the convex end.

□ Contour Plots of the Shape Functions

The following code calls the shape function generator, **p** is the list of the nodal coordinates, and the physical domain is represented by the **x-y** frame. An example with the following nodes:

```
In[23]:= p = {{0, 0}, {-1.5, -0.5}, {1, 0}, {0.5, 0.5}}
```

```
Out[23]= {{0, 0}, {-1.5, -0.5}, {1, 0}, {0.5, 0.5}}
```

is presented here.

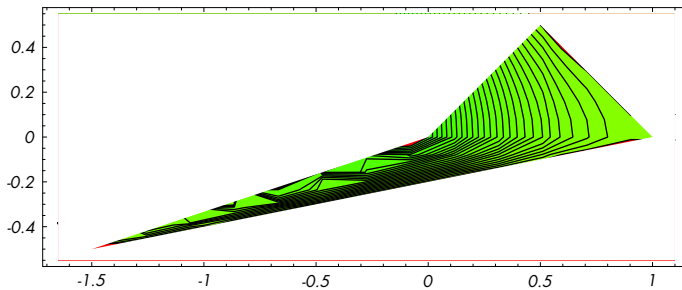
```
In[24]:= (* FourShapes[p,{x,y}] *)
```

The linearity condition along the straight boundary segments of the calculated shape functions is verified:

$$\text{Shapes along sides: } \begin{pmatrix} 1. + 0.666667 x & -0.666667 x & 0 & 0 \\ 0 & 0.4 - 0.4 x & 0.6 + 0.4 x & 0 \\ 0 & 0 & -1. + 2. x & 2. - 2. x \\ 1. - 2. x & 0 & 0 & 2. x \end{pmatrix}$$

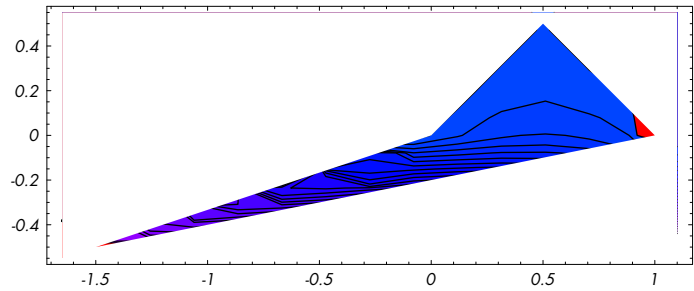
In the matrix, rows indicate the shape function and columns indicate corresponding sides of the quadrilateral.

The original shape function is displayed with unit value at the concave end and with zero values at the remaining ends.

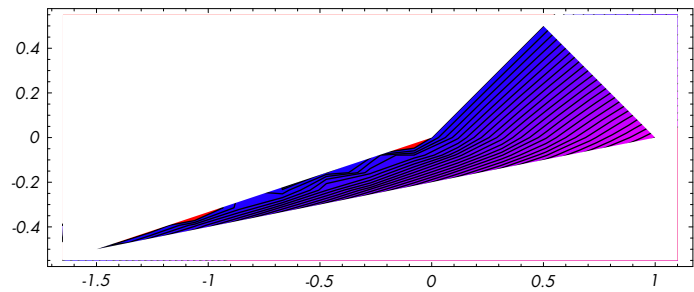


The following three shape functions are calculated from the linear interpolating condition: **f={1,x,y}**.

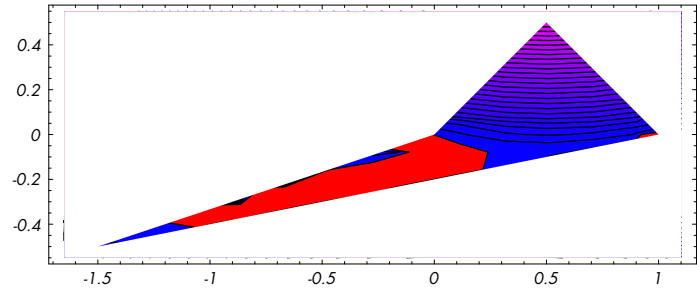
The second shape function below is associated with the bottom-most node.



The third interpolant shown below corresponds to the node facing the concave end.



The following interpolant is associated with the top-most node. Very low values can be observed in the bottom region of the element.



The following analytical forms of the shape functions can be used to generate the element stiffness matrix using the exact integration *Mathematica* code presented at *IAS* in 1998, in Murmansk, Russia.

$$\left\{ \begin{aligned} & \{(-1. + x)^2 - 1. y^2, -0.333333 (-1. + x) x - 0.333333 y + 0.333333 y^2, \\ & -0.333333 (-4. + x) x - 1.333333 y + 0.333333 y^2, \\ & -0.333333 (-1. + x) x + 1.66667 y + 0.333333 y^2\}, \\ & \{-9. (-0.555556 + y) (0.2 + y) + x \left(-1.36 + x + \frac{0.128 x}{-0.2 x + y}\right), \\ & y (-1.4 + 3. y) + \frac{x (0.0666667 (-1. + x) x - 0.333333 (-0.36 + x) y)}{-0.2 x + y}, \\ & y (-2.4 + 3. y) + \frac{x (0.0666667 (-4. + x) x - 0.333333 (-3.36 + x) y)}{-0.2 x + y}, \\ & y (0.6 + 3. y) + \frac{x (0.0666667 (-1. + x) x - 0.333333 (-0.36 + x) y)}{-0.2 x + y} \} \end{aligned} \right\}$$

Observe that the Padé form is automatically captured.

□ Closed-Form Result

Let us consider a quadrilateral with the following nodes.

```
In[25]:= p = {{x1, y1}, {x2, y2}, {x3, y3}, {x4, y4}}
```

```
Out[25]:= {{x1, y1}, {x2, y2}, {x3, y3}, {x4, y4}}
```

Let the shape function associated with $\{x_1, y_1\}$ be $n_1[x, y]$ in the x - y coordinate frame. The three other shape functions, $n_2[x, y]$, $n_3[x, y]$, and $n_4[x, y]$ were obtained from:

```
In[26]:= Shape1ToAll[n1xy_,
  {{x1_, y1_}, {x2_, y2_}, {x3_, y3_}, {x4_, y4_}}] :=
Module[{n1}, {n1, ({-(x3 y - x4 y - n1 x3 y1 + n1 x4 y1 - x y3 + n1 x1 y3 +
  x4 y3 - n1 x4 y3 + x y4 - n1 x1 y4 - x3 y4 + n1 x3 y4),
  x2 y - x4 y - n1 x2 y1 + n1 x4 y1 - x y2 + n1 x1 y2 + x4 y2 - n1 x4 y2 +
  x y4 - n1 x1 y4 - x2 y4 + n1 x2 y4, -x2 y + x3 y + n1 x2 y1 - n1 x3 y1 +
  x y2 - n1 x1 y2 - x3 y2 + n1 x3 y2 - x y3 + n1 x1 y3 + x2 y3 - n1 x2 y3} /
  Det[{{1, x2, y2}, {1, x3, y3}, {1, x4, y4}}])
} /. n1 -> n1xy];
```

A remarkable result is that if $n_1[x, y]$ satisfies the elliptic condition $D[n_1, \{x, 2\}] + \alpha[x, y] D[n_1, \{y, 2\}] == 0$, then the same partial differential equation, for arbitrary $\alpha[x, y] > 0$, is satisfied by the three computed shape functions as verified below.

```
In[27]:= (Simplify[( $\alpha$  D[#, {x, 2}] + D[#, {y, 2}]) /.
  n1(0,2)[x, y] -> - $\alpha$  n1(2,0)[x, y]]) & /@ {n2, n3, n4}
```

```
Out[27]:= {0, 0, 0}
```

■ Conclusions

The biomedical application of maxillo-facial morphometric analysis can be cited as an important application of the present methodology. The idea is now being utilized for general three-dimensional cases. For the unconventional concave

quadrilateral finite element, the proposed formulation made it possible to generate all shape functions starting from the one which pertains to the concave node.

To our knowledge there is neither any published work nor any hint that the shape functions for elements can be generated from a single one by satisfying (desirable) interpolation conditions. Perhaps no investigator other than Wachspress could conceive the shape functions in the $\mathbf{x}\text{-}\mathbf{y}$ frame. In fact, except for Taig's isoparametric formulation in the computational domain of a unit square, there exists no other original idea, to our knowledge, of generating shape functions. For the quadrilateral example presented here, it has been shown that if the first shape function is generated by isoparametric formulation, then the three remaining functions can be constructed according to the proposed scheme, automatically satisfying the patch test.

■ References

- [1] M. L. Moss, and L. Salentijn, "The Primary Role of the Functional Matrix in Facial Growth," *The Angle Orthodontist*, **55**, 1969 pp. 566–574.
- [2] www.wolfram.com/products/applications/parallel
- [3] www.wolfram.com/products/applications/mathcode
- [4] P. Planche and J. Treil, "Cephalometrics for the Next Millennium," In *Proceedings: The Future of Orthodontics*, Leuven University Press, France, 1998 pp. 181–192.
- [5] J. Treil, C. Madrid, M. Jaeger, J. Casteigt, Ph. Borianne, "Biometrie Tridimensionnelle Maxillo-Faciale," *Biometrie Humaine et Anthropologie*, France, **15**(1.2), 1997 pp. 65–73.
- [6] I. C. Taig, "Structural Analysis by the Matrix Displacement Method," *English Electric Aviation*, Report Number S017, 1961.
- [7] E. L. Wachspress, *A Rational Finite Element Basis*, New York: Academic Press, 1975.
- [8] M. Laydi and P. Lesaint, "Sur les Éléments Finis Rationnels de Serendip de Degré Quelconque," *Numerische Mathematik*, Springer Verlag, 1985 pp. 175–187.
- [9] O. C. Zienkiewicz and R. L. Taylor, 3rd ed., *The Finite Element Method*, London; New York: McGraw-Hill, 1977.
- [10] J. L. Gout, "Construction of an Hermite Rational Interpolation Wachspress Type Finite Element," *Journal of Computational Applied Mathematics*, **5**, 1979 pp. 337–347.

■ Acknowledgment

Gautam Dasgupta was supported by a grant from the National Science Foundation, cms-9820353.

■ Additional Material

Available at www.mathematica-journal.com.

Appendices.nb

About the Authors

Gautam Dasgupta is a professor of Civil Engineering and Engineering Mechanics at the Columbia University, New York. His areas of research include computer mathematics, waves in random media, earthquake engineering, and finite and boundary element analysis with *Mathematica*. For details on his recent work, see www.columbia.edu/~gd18.

Doctor Jacques Treil is a Neuro and Maxillo-facial Radiologist and a full member of the French Society of Neuroradiology, the European Society of Neuroradiology, and the French Orthodontics Society.

Theorema: A Proving System Based on Mathematica

Bruno Buchberger

*Research Institute for Symbolic Computation
Johannes Kepler University Linz
A4232 Schloss Hagenberg, Austria
theorema@theorema.org
www.theorema.org*

■ What is Theorema?

I have a passion for proving. In my view, proving is the essence of mathematics. As soon as we ask “why?” we start proving. Proving is cultivated reasoning. Mathematics cannot be defined by the objects studied but only by its method, which is proving. Formal proving, the basis of computer-supported proving, will drastically change the way we do mathematics. Computing is a special case of proving, and proving is a special case of computing. Actually, computing and proving are two of the three main aspects of formal mathematical activity. The third aspect is solving. Proving, solving, and computing naturally correspond to the way free variables in formulae are considered: They can be universally quantified, existentially quantified, or left free for substitution.

The Theorema system is programmed in *Mathematica* and heavily uses advanced features of the kernel and the front end of *Mathematica*. Theorema is developed by the Theorema Working Group (see www.theorema.org). Theorema allows users to compose “active” formal mathematical texts and to process them by proving, solving, and computing (simplifying). Theorema is based on a version of higher order predicate logic and, thus, is a possible frame for all of mathematics.

Originally presented at the Third International *Mathematica* Symposium (Hagenberg, Austria, August 23–25, 1999).

Some of the unique features of Theorema include the following.

- The fundamental interaction between proving, solving, and computing is made explicit.
- New facilities for automated proving are provided, and, at the same time, full access to the solving and computing power of *Mathematica* is given.
- The “usual” mathematical syntax and consistent semantics is implemented, and proofs are generated in a “natural” style.
- Theorema has *one* uniform logic and software frame, but offers *many* general *and* special proving, solving, and simplifying methods.

■ Theorema’s Prove Command

The current provers of Theorema include various predicate logic provers, various induction provers, a set theory prover, and special provers using the Gröbner bases technique, Collins quantifier elimination, and the Zeilberger-Paule approach to handling combinatorial identities. A recent proof method developed by the author, called the “PCS-method”, is particularly well-suited for proving formulae with “alternating quantifiers”, which typically appear in proofs in analysis. An example of applying this method is given below.

Theorema also includes special tools for mathematical knowledge management, natural language proof presentation, and the generation of new pictorial symbols, called “logicographic symbols”.

Here is a typical example of defining a mathematical notion in Theorema by entering it in a *Mathematica* input cell.

```
Definition["limit:", any[f, a],
  limit[f, a]  $\Leftrightarrow \forall_{\epsilon > 0} \exists_{N \in \mathbb{N}} \forall_{n \geq N} |f[n] - a| < \epsilon$ 
```

For `limit[f, a]`, read “the sequence f has limit a ”. Now, one may be interested in proving a proposition on this notion, such as the following one.

```
Proposition["limit of sum", any[f, a, g, b],
  (limit[f, a]  $\wedge$  limit[g, b])  $\Rightarrow$  limit[f + g, a + b]]
```

Of course, this proposition cannot be proved without any knowledge of auxiliary notions like ‘+’ and so forth. It may be surprising that actually very little knowledge is necessary for proving the proposition. In this example, the following statements are sufficient for the proof.

```
Definition["+":, any[f, g, x],
  (f + g)[x] = f[x] + g[x]]

Lemma["|+|", any[x, y, a, b,  $\delta$ ,  $\epsilon$ ],
  (|(x + y) - (a + b)| < ( $\delta$  +  $\epsilon$ ))  $\Leftarrow$  (|x - a| <  $\delta \wedge$  |y - b| <  $\epsilon$ )]

Lemma["max", any[m, M1, M2],
  m  $\geq$  max[M1, M2]  $\Rightarrow$  (m  $\geq$  M1  $\wedge$  m  $\geq$  M2)]
```

In Theorema, formulae can be combined in “knowledge bases” using the “Theory” construct, which should be self-explanatory.

```

Theory["limit",
  Definition["limit:"]
  Definition["+:" ]
  Lemma["|+|"]
  Lemma["max"]

```

Now a proof of the above goal proposition can be generated completely automatically by just entering the following.

```

Prove[Proposition["limit of sum"], using → Theory["limit"], by → PCS]

-ProofObject-

```

The three main arguments of a call to the Theorema command **Prove** are: the name of the proposition to be proved, the name of the theory (knowledge base) to be used in the proof, and the particular prover by which the proof should be generated. After entering this command, the proof shown in the next section, including the English explanatory intermediate text, will be generated in an extra notebook without any further user interaction.

■ A Proof Generated by Theorema

Prove:

$$\begin{aligned}
 \text{(Proposition (limit of sum))} \quad & \forall_{f,a,g,b} (\text{limit}[f, a] \wedge \text{limit}[g, b] \Rightarrow \\
 & \text{limit}[f+g, a+b]),
 \end{aligned}$$

under the assumptions:

$$\text{(Definition (limit:))} \quad \forall_{f,a} \left(\text{limit}[f, a] \Leftrightarrow \forall_{\epsilon} \exists_{N} \forall_{n \geq N} (|f[n] - a| < \epsilon) \right),$$

$$\text{(Definition (+:))} \quad \forall_{f,g,x} ((f+g)[x] = f[x] + g[x]),$$

$$\begin{aligned}
 \text{(Lemma (|+|))} \quad & \forall_{x,y,a,b,\delta,\epsilon} (|(x+y) - (a+b)| < \delta + \epsilon \Leftrightarrow \\
 & (|x-a| < \delta \wedge |y-b| < \epsilon)),
 \end{aligned}$$

$$\text{(Lemma (max))} \quad \forall_{m,M1,M2} (m \geq \max[M1, M2] \Rightarrow m \geq M1 \wedge m \geq M2).$$

We assume

$$(1) \quad \text{limit}[f_0, a_0] \wedge \text{limit}[g_0, b_0],$$

and show

$$(2) \quad \text{limit}[f_0 + g_0, a_0 + b_0].$$

Formula (1.1), by (Definition (limit:)), implies:

$$(3) \quad \forall_{\epsilon > 0} \exists_N \forall_n \quad (|f_0[n] - a_0| < \epsilon).$$

By (3), we can take an appropriate Skolem function such that

$$(4) \quad \forall_{\epsilon > 0} \forall_n \quad (|f_0[n] - a_0| < \epsilon),$$

Formula (1.2), by (Definition (limit:)), implies:

$$(5) \quad \forall_{\epsilon > 0} \exists_N \forall_n \quad (|g_0[n] - b_0| < \epsilon).$$

By (5), we can take an appropriate Skolem function such that

$$(6) \quad \forall_{\epsilon > 0} \forall_n \quad (|g_0[n] - b_0| < \epsilon),$$

Formula (2), using (Definition (limit:)), is implied by:

$$(7) \quad \forall_{\epsilon > 0} \exists_N \forall_n \quad (|(f_0 + g_0)[n] - (a_0 + b_0)| < \epsilon).$$

We assume

$$(8) \quad \epsilon_0 > 0,$$

and show

$$(9) \quad \exists_N \forall_n \quad (|(f_0 + g_0)[n] - (a_0 + b_0)| < \epsilon_0).$$

We have to find N^* such that

$$(10) \quad \forall_n \quad (n \geq N^* \Rightarrow |(f_0 + g_0)[n] - (a_0 + b_0)| < \epsilon_0).$$

Formula (10), using (Definition (+:)), is implied by:

$$(11) \quad \forall_n \quad (n \geq N^* \Rightarrow |(f_0[n] + g_0[n]) - (a_0 + b_0)| < \epsilon_0).$$

Formula (11), using (Lemma (|+|)), is implied by:

$$(12) \quad \exists_{\delta, \epsilon} \forall_n \quad (n \geq N^* \Rightarrow |f_0[n] - a_0| < \delta \wedge |g_0[n] - b_0| < \epsilon).$$

We have to find δ^* , ϵ^* and N^* such that

$$(13) \quad (\delta^* + \epsilon^* = \epsilon_0) \bigwedge_n \quad (n \geq N^* \Rightarrow |f_0[n] - a_0| < \delta^* \wedge |g_0[n] - b_0| < \epsilon^*).$$

Formula (13), using (6), is implied by:

$$(\delta^* + \epsilon^* = \epsilon_0) \bigwedge_n \quad (n \geq N^* \Rightarrow \epsilon^* > 0 \wedge n \geq N_1[\epsilon^*] \wedge |f_0[n] - a_0| < \delta^*),$$

which, using (4), is implied by:

$$(\delta^* + \epsilon^* = \epsilon_0) \bigwedge_n \quad (n \geq N^* \Rightarrow \delta^* > 0 \wedge \epsilon^* > 0 \wedge n \geq N_0[\delta^*] \wedge n \geq N_1[\epsilon^*]),$$

which, using (Lemma (max)), is implied by:

$$(14) \quad (\delta^* + \epsilon^* = \epsilon_0) \bigwedge_n (n \geq N^* \Rightarrow \delta^* > 0 \wedge \epsilon^* > 0 \wedge n \geq \max[N_0[\delta^*], N_1[\epsilon^*]]).$$

Formula (14) is implied by:

$$(15) \quad (\delta^* + \epsilon^* = \epsilon_0) \bigwedge \delta^* > 0 \bigwedge \epsilon^* > 0 \bigwedge_n (n \geq N^* \Rightarrow n \geq \max[N_0[\delta^*], N_1[\epsilon^*]]).$$

Partially solving it, formula (15) is implied by:

$$(16) \quad (\delta^* + \epsilon^* = \epsilon_0) \wedge \delta^* > 0 \wedge \epsilon^* > 0 \wedge (N^* = \max[N_0[\delta^*], N_1[\epsilon^*]]).$$

Now,

$$(\delta^* + \epsilon^* = \epsilon_0) \wedge \delta^* > 0 \wedge \epsilon^* > 0$$

can be solved for δ^* and ϵ^* by a call to Collins' CAD method yielding the solution

$$0 < \delta^* < \epsilon_0,$$

$$\epsilon^* \leftarrow \epsilon_0 - 1 * \delta^*.$$

Let us take

$$N^* \leftarrow \max[N_0[\delta^*], N_1[\epsilon_0 - 1 * \delta^*]].$$

Formula (16) is solved. Hence, we are done.

■ Perspective

Theorema provides a common frame for proving, solving, and simplifying (computing). More and more provers, solvers, and simplifiers are currently added to Theorema and links to other proving systems are provided from within Theorema. The user can introduce new mathematical notions and experiment with the notions by computing examples. He can formulate conjectures and try to prove them by using the provers from the Theorema prover library. He can also study the question under which assumptions a given conjecture is true, and he can compose theories as hierarchical knowledge bases. Nonalgorithmic theorems and algorithms are expressed in one common language frame. For structuring towers of algorithmic mathematical domains, Theorema also provides the functor construct.

While, in the present version, the prover library of Theorema cannot be changed by the user, the next version of Theorema is intended to provide language constructs that will allow the user to create, with little effort, new provers, solvers, and simplifiers from the ones provided in the Theorema library.

■ Acknowledgment

The Theorema project is sponsored by Project SFB 1302 of the Austrian Science Foundation (FWF) and Project Prove of the Upper Austrian Government.

About the Author

Bruno Buchberger is professor of computer mathematics at the Research Institute for Symbolic Computation (RISC) of the Johannes Kepler University in Linz, Austria. He is the founder of RISC and the founding editor of the Journal of Symbolic Computation. His main research contribution is the theory of Gröbner bases. His current research interest is the Theorema Project.

Code Generation for Simulation and Control Applications

Mats Jirstrand and Johan Gunnarsson

MathCore AB

Wallenbergs gata 4

SE-583 35 Linköping, Sweden

The use of *Mathematica* in combination with *MathCode C++* is illustrated in a context of modeling of dynamical systems and design of controllers. The symbolic tools are used to derive a set of nonlinear differential equations using Euler–Lagrange equations of motion. The model is converted to C++ using *MathCode C++*, which produces an efficient implementation of the large expressions used in the model. The exported code is used for simulations, which illustrates that *Mathematica* in combination with *MathCode C++* can be used to do accurate and powerful simulations of nonlinear systems. Controller synthesis is performed where the resulting controller is exported to C++ and run externally. The applications presented are a seesaw/pendulum process and the aerodynamics of a fighter aircraft.

■ 1. Introduction

Control system design is a discipline where advanced mathematics is applied to real-world problems ranging from pulp and paper manufacturing, aircraft, and CD players to biochemical processes, logistics, and financial applications. *Mathematica* is very powerful for dealing with the mathematics in these control problems, and the application package *Control System Professional* (CSP) implements many useful features. When it comes to nonlinear systems, the combination of the symbolic and numeric capabilities of *Mathematica* makes modeling and control system design in this environment very attractive. The notebook concept for documentation of different tradeoffs and decisions during the design process also contributes to making *Mathematica* suitable for this kind of work.

Originally presented at the Third International *Mathematica* Symposium (Hagenberg, Austria, August 23–25, 1999).

The application package *MathCode C++* adds automatic C++ code generation to *Mathematica*, which can be used both to enhance the performance of simulations of large systems and to generate standalone C++ code to be used in applications separately from *Mathematica*. Performance will be the issue in the seesaw/pendulum example below while the standalone feature is explored in the fighter aircraft example. The standalone code generation feature of *MathCode C++* makes it possible to design and prototype a controller in *Mathematica* and then “lift out” the resulting standalone code to be used in the real control system. This minimizes the need to code the control law manually from the algorithms designed in *Mathematica*.

In this document we will try to illustrate the use of *Mathematica* together with *MathCode C++* for modeling, control system design, and code generation. The document is organized as follows. In the rest of this section we give some basic facts about dynamic systems needed to formulate controller design problems. In section 2, a nonlinear mechanical system is modeled and simulated using generated C++ code. In section 3, controller design for an aircraft is done, and in section 4, we give some conclusions.

□ Preliminaries

Many dynamic systems can be modeled by a set of first-order differential equations. See, for example, [1, 2]

$$\begin{aligned}\frac{dx[t]}{dt} &= f[x[t], u[t]] \\ y[t] &= g[x[t]]\end{aligned}\tag{1}$$

where $f: \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n$. Here $x[t] \in \mathbb{R}^n$, $u[t] \in \mathbb{R}^m$, and $y[t] \in \mathbb{R}^p$ are called the *states*, *inputs*, and *outputs* of the system, respectively. Equation (1) is called the *state space form* of the equations describing the behavior of the system and is particularly useful for control system design. We will also use the dot-notation $\dot{x}$ for denoting differentiation with respect to time $\frac{dx}{dt}$. If the system is linear, the state space form can be written as

$$\begin{aligned}\frac{dx[t]}{dt} &= \mathcal{A} \cdot x[t] + \mathcal{B} \cdot u[t], \\ y[t] &= \mathcal{C} \cdot x[t]\end{aligned}\tag{2}$$

where $\mathcal{A} \in \mathbb{R}^{n \times n}$, $\mathcal{B} \in \mathbb{R}^{n \times m}$, and $\mathcal{C} \in \mathbb{R}^{p \times n}$ are constant real matrices. If $f[0, 0] = 0$ in equation (1), the system can be linearized around $x = 0$, $u = 0$. The matrices $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ are then computed from f and g by taking partial derivatives as follows.

$$\begin{aligned}\mathcal{A} &= \left. \frac{\partial f[x, u]}{\partial x} \right|_{x=0, u=0}, \\ \mathcal{B} &= \left. \frac{\partial f[x, u]}{\partial u} \right|_{x=0, u=0}, \\ \mathcal{C} &= \left. \frac{\partial g[x]}{\partial x} \right|_{x=0}\end{aligned}\tag{3}$$

The linearized model is usually a good approximation whenever the states and inputs to the system are small. There exists a large number of control design methods for linear systems that makes it desirable to work with linear models if possible during controller synthesis. The performance of the resulting controller can then be studied in simulations where the linear model has been exchanged with the nonlinear one.

A linear state-feedback controller is a very simple type of controller where the control signal is a linear combination of the states that are assumed to be measurable. Hence, a linear state-feedback law has the form

$$u[t] = -\mathcal{L} \cdot x[t] \quad (4)$$

There exist many methods for computing the matrix $\mathcal{L}$ but a necessary condition is that the chosen $\mathcal{L}$ makes the closed-loop system asymptotically stable, which essentially means that nonzero states converge to zero. More or less advanced methods make it possible to compute $\mathcal{L}$ such that different tradeoffs between performance and robustness can be obtained; see, for example, [1, 3, 4].

A slightly more advanced controller is the linear dynamic controller, which includes an internal model of the system to be controlled. A so-called observer is used to estimate the states $x[t]$ of the system from measured signals $y[t]$ and a linear feedback law is used to compute the input to the system to be controlled. Using the notation $\hat{x}[t]$ for estimated states, the linear dynamic controller can be written as follows.

$$\begin{aligned} \frac{d\hat{x}[t]}{dt} &= \mathcal{A}_c \cdot \hat{x}[t] + \mathcal{B}_c \cdot y[t], \\ u[t] &= -\mathcal{L} \cdot \hat{x}[t] \end{aligned} \quad (5)$$

Hence, a dynamic controller requires real-time solving of a system of differential equations. In the seesaw/pendulum example below, we will use a linear state-feedback controller of the form in equation (1), and in the fighter aircraft example, we will use the observer-based controller of the form in equation (5).

■ 2. The Seesaw/Pendulum Process

In this section, we will illustrate how the symbolic capabilities of *Mathematica* can be used to derive a model of a nonlinear system. Using *MathCode C++*, the large symbolic expressions in the nonlinear model are then converted to C++ code and used to simulate the system very efficiently. The system is a laboratory process frequently used in control education known as the seesaw/pendulum process. One implementation of the seesaw/pendulum process has been developed by Quanser Consulting (www.quanser.com).

□ The System

The process consists of a seesaw, two carts called C_1 and C_2 , two parallel tracks, an inverted pendulum, and a weight. Each cart can be driven by a DC motor

controlled by an input voltage. Cart C_1 carries the weight and cart C_2 carries the inverted pendulum attached by a friction-free joint. The carts can be moved along the tracks by controlling the input voltages of the DC motors.

We start by modeling the open loop system, that is, without any feedback from measured signals. The forces F_1 and F_2 acting on each cart are chosen as inputs. We will use the Lagrangian methodology to obtain a nonlinear model of the system and then linearize it. The linearized model is used for control system design where the *Control System Professional* application package is used. The closed-loop system is then simulated both within *Mathematica* and by external code generated using *MathCode C++*.

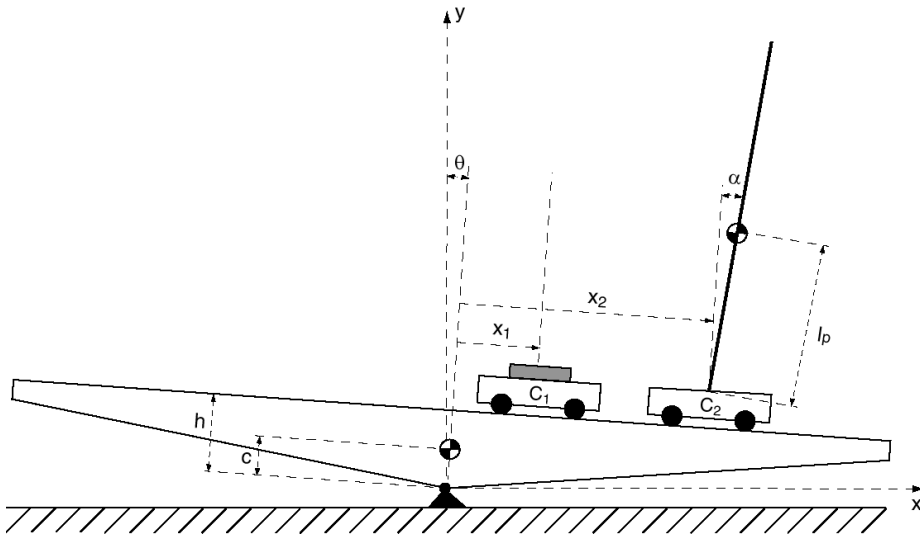


Figure 1. The pendulum/seesaw process.

In Figure 1 we introduce names of variables and constants describing the pendulum/seesaw process.

Variables:

x_1	translation of cart 1 from center of track
x_2	translation of cart 2 from center of track
θ	angle of seesaw with vertical
α	angle of inverted pendulum with normal to track
F_1	force applied to cart 1
F_2	force applied to cart 2

Constants:

J	inertia of seesaw with track at height h
m_s	mass of seesaw with track
c	height of center of gravity of the seesaw from pivot point
h	height of track from pivot point
m_1	mass of cart 1 (weight cart, on the back track)
m_2	mass of cart 2 (pendulum cart, on the front track)

m_p	mass of pendulum
l_p	center of mass of pendulum rod (half of full length)
g	gravitational acceleration

The physical values of the above constants are stored as rules in *Mathematica* to be used in later simulations.

$$\text{physicalvalues} := \left\{ J \rightarrow 1.6, m_s \rightarrow 6.6, c \rightarrow 0.06, h \rightarrow 0.115, \right. \\ \left. m_1 \rightarrow 0.48 + 0.38, m_2 \rightarrow 0.48, m_p \rightarrow 0.2, l_p \rightarrow \frac{0.61}{2} + 0.03, g \rightarrow 9.81 \right\}$$

□ Modeling

Following the Lagrangian methodology [5], the system is divided into a number of subsystems whose potential energy and kinetic energy are computed in terms of the generalized coordinates introduced in Figure 1. The Lagrangian, which is the difference between the total kinetic and potential energy, can then be used to derive the equations of motion for the system.

Computation of the Lagrangian

Coordinates of the center of mass of the seesaw:

$$\begin{aligned} x_s &:= c \sin[\theta[t]] \\ y_s &:= c \cos[\theta[t]] \end{aligned}$$

The potential and kinetic energies of the seesaw:

$$\begin{aligned} V_s &:= m_s g y_s \\ T_s &:= \text{Simplify}\left[\frac{1}{2} J (\partial_t \theta[t])^2\right] \end{aligned}$$

Coordinates of the center of track:

$$\begin{aligned} x_c &:= h \sin[\theta[t]] \\ y_c &:= h \cos[\theta[t]] \end{aligned}$$

Coordinates of cart 1:

$$\begin{aligned} x_{m1} &:= x_c + x_1[t] \cos[\theta[t]] \\ y_{m1} &:= y_c - x_1[t] \sin[\theta[t]] \end{aligned}$$

The potential and kinetic energies of cart 1:

$$\begin{aligned} V_{m1} &:= m_1 g y_{m1} \\ T_{m1} &:= \text{Simplify}\left[\frac{1}{2} m_1 ((\partial_t x_{m1})^2 + (\partial_t y_{m1})^2)\right] \end{aligned}$$

Coordinates of cart 2:

$$\begin{aligned} x_{m2} &:= x_c + x_2[t] \cos[\theta[t]] \\ y_{m2} &:= y_c - x_2[t] \sin[\theta[t]] \end{aligned}$$

The potential and kinetic energies of cart 2:

$$\begin{aligned} V_{m2} &:= m_2 g y_{m2} \\ T_{m2} &:= \text{Simplify}\left[\frac{1}{2} m_2 ((\partial_t x_{m2})^2 + (\partial_t y_{m2})^2)\right] \end{aligned}$$

Coordinates of the center of mass of the pendulum:

$$\begin{aligned} \mathbf{x}_p &:= \mathbf{x}_{m2} + l_p \sin[\alpha[t] + \theta[t]] \\ \mathbf{y}_p &:= \mathbf{y}_{m2} + l_p \cos[\alpha[t] + \theta[t]] \end{aligned}$$

The potential and kinetic energies of the pendulum:

$$\begin{aligned} V_p &:= m_p g y_p \\ T_p &:= \text{Simplify}\left[\frac{1}{2} m_p ((\partial_t x_p)^2 + (\partial_t y_p)^2)\right] \end{aligned}$$

The total potential energy:

$$V_{\text{tot}} := V_s + V_{m1} + V_{m2} + V_p$$

The total kinetic energy:

$$T_{\text{tot}} := T_s + T_{m1} + T_{m2} + T_p$$

The Lagrangian of the system:

$$L := T_{\text{tot}} - V_{\text{tot}}$$

becomes

$$\begin{aligned} &\text{Simplify}[L] \\ &\frac{1}{2} (-2 c g \cos[\theta[t]] m_s - 2 g m_1 (h \cos[\theta[t]] - \sin[\theta[t]] x_1[t]) - \\ &\quad 2 g m_2 (h \cos[\theta[t]] - \sin[\theta[t]] x_2[t]) - \\ &\quad 2 g m_p (h \cos[\theta[t]] + \cos[\alpha[t] + \theta[t]] l_p - \sin[\theta[t]] x_2[t]) + \\ &\quad J \theta'[t]^2 + m_1 ((h^2 + x_1[t]^2) \theta'[t]^2 + 2 h \theta'[t] x_1'[t] + x_1'[t]^2) + \\ &\quad m_2 ((h^2 + x_2[t]^2) \theta'[t]^2 + 2 h \theta'[t] x_2'[t] + x_2'[t]^2) + \\ &\quad m_p ((h \cos[\theta[t]] - \sin[\theta[t]] x_2[t]) \theta'[t] + \\ &\quad \cos[\alpha[t] + \theta[t]] l_p (\alpha'[t] + \theta'[t]) + \cos[\theta[t]] x_2'[t])^2 + \\ &\quad ((h \sin[\theta[t]] + \cos[\theta[t]] x_2[t]) \theta'[t] + \sin[\alpha[t] + \theta[t]] \\ &\quad l_p (\alpha'[t] + \theta'[t]) + \sin[\theta[t]] x_2'[t])^2)) \end{aligned}$$

Computation of the Equations of Motion

The equations of motion are given by the partial differential equations that L must satisfy

$$\frac{d}{dt} \frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} = Q_i, \quad i = 1, \dots, n, \quad (6)$$

where L is the Lagrangian, q_i , $i = 1, \dots, n$ are the generalized coordinates, and Q_i , $i = 1, \dots, n$ are the generalized forces associated with each coordinate. In this case, the quadruple $\{q_1, q_2, q_3, q_4\}$ corresponds to $\{x_1, x_2, \theta, \alpha\}$ and $Q_1 = F_1$, $Q_2 = F_2$, $Q_3 = Q_4 = 0$. We derive each equation and simplify it.

$$\begin{aligned} \text{eqn}_1 &= \text{Simplify}[\partial_t (\partial_{\partial_t x_1[t]} L) - \partial_{x_1[t]} L == F_1]; \\ \text{eqn}_2 &= \text{Simplify}[\partial_t (\partial_{\partial_t x_2[t]} L) - \partial_{x_2[t]} L == F_2]; \\ \text{eqn}_3 &= \text{Simplify}[\partial_t (\partial_{\partial_t \theta[t]} L) - \partial_{\theta[t]} L == 0]; \\ \text{eqn}_4 &= \text{Simplify}[\partial_t (\partial_{\partial_t \alpha[t]} L) - \partial_{\alpha[t]} L == 0]; \end{aligned}$$

These equations are coupled ordinary differential equations (ODEs) of second-order in the generalized coordinates x_1 , θ , x_2 , α . To rewrite these in standard state-space form (a system of first-order ODEs) we introduce the following states:

$$x = (x_1 \ \theta \ x_2 \ \alpha \ \dot{x}_1 \ \dot{\theta} \ \dot{x}_2 \ \dot{\alpha})$$

The inputs to the system are

$$u = (F_1 \ F_2)$$

We observe that the second-order time derivatives of the positions and angles appear linearly, which makes it easy to solve for these in terms of the states. Second-order time derivatives will *always* appear linearly in the equations derived from the partial differential equation (6) that the Lagrangian of the system has to satisfy. This is due to the fact that the Lagrangian only consists of at most first-order time derivatives and the second-order derivatives appear according to the chain rule when differentiating the term $\frac{\partial L}{\partial \dot{q}_i}$ in equation (6) with respect to time.

Solve for second-order derivatives.

```
sol = Solve[ { eqn1, eqn2, eqn3, eqn4 },
  {  $\partial_{\{t,2\}} x_1[t]$ ,  $\partial_{\{t,2\}} x_2[t]$ ,  $\partial_{\{t,2\}} \theta[t]$ ,  $\partial_{\{t,2\}} \alpha[t]$  } ];
```

These solutions will become a part of the right-hand sides of the differential equations used for simulating the system.

We introduce conversion rules to get rid of explicit time dependence. This makes some expressions look simpler and takes less space.

```
ssvariables := { x1[t] → x1,  $\theta[t]$  →  $\theta$ , x2[t] → x2,  $\alpha[t]$  →  $\alpha$ ,
   $\partial_t x_1[t]$  → dx1,  $\partial_t \theta[t]$  → d $\theta$ ,  $\partial_t x_2[t]$  → dx2,  $\partial_t \alpha[t]$  → d $\alpha$  };
```

We compute the right-hand side of the nonlinear state-space form (equation (1)) of the equation

$$\dot{x}[t] = f[x[t], u[t]]$$

which becomes very large! Observe that the first four entries in f correspond to pure integrations. This allow us to write the model as a system of first-order differential equations as desired.

$$f = \begin{pmatrix} \partial_t x_1[t] \\ \partial_t \theta[t] \\ \partial_t x_2[t] \\ \partial_t \alpha[t] \\ \partial_{\{t,2\}} x_1[t] \\ \partial_{\{t,2\}} \theta[t] \\ \partial_{\{t,2\}} x_2[t] \\ \partial_{\{t,2\}} \alpha[t] \end{pmatrix} /. \text{sol} /. \text{ssvariables} // \text{Flatten};$$

Short[%, 10]

$$\left\{ dx_1, d\theta, dx_2, \langle\langle 3 \rangle\rangle, \right. \\ \left. - \frac{\text{Sec}[\alpha] (\langle\langle 1 \rangle\rangle)}{l_p m_p} + \frac{\langle\langle 1 \rangle\rangle}{\langle\langle 1 \rangle\rangle} + \frac{\text{Sec}[\alpha] \langle\langle 1 \rangle\rangle \left(\frac{\langle\langle 1 \rangle\rangle - \langle\langle 1 \rangle\rangle}{\langle\langle 1 \rangle\rangle - \langle\langle 1 \rangle\rangle} - \langle\langle 1 \rangle\rangle \right)}{l_p m_p}, \right. \\ \left. - \left((\text{Cos}[\alpha] l_p m_p (-F_2 - g \text{Sin}[\theta] m_2 - g \text{Sin}[\theta] m_p - d\alpha^2 \text{Sin}[\alpha] l_p m_p - \right. \right. \\ \left. \left. 2 d\alpha d\theta \text{Sin}[\alpha] l_p m_p - d\theta^2 \text{Sin}[\alpha] l_p m_p - d\theta^2 m_2 x_2 - d\theta^2 m_p x_2) - \right. \right. \\ \left. \left. (m_2 + m_p) (-g \text{Sin}[\alpha + \theta] l_p m_p + 2 d\theta \text{Sin}[\alpha] dx_2 l_p m_p + \right. \right. \\ \left. \left. d\theta^2 l_p m_p (h \text{Sin}[\alpha] - \text{Cos}[\alpha] x_2)) \right) \right. \\ \left. (\langle\langle 1 \rangle\rangle) + (\langle\langle 1 \rangle\rangle - (m_2 + m_p) (\langle\langle 1 \rangle\rangle)) (\langle\langle 1 \rangle\rangle) / \right. \\ \left. (J \text{Cos}[\alpha] l_p^3 m_1 m_2 m_p^2 + J \text{Cos}[\alpha] l_p^3 m_1 m_p^3 - J \text{Cos}[\alpha]^3 l_p^3 m_1 m_p^3 + \right. \\ \left. \text{Cos}[\alpha] l_p^3 m_1^2 m_2 m_p^2 x_1^2 + \langle\langle 9 \rangle\rangle + \text{Cos}[\alpha] l_p^3 m_1 m_p^4 x_2^2 - \right. \\ \left. \text{Cos}[\alpha]^3 l_p^3 m_1 m_p^4 x_2^2 - \text{Cos}[\alpha] \text{Sin}[\alpha]^2 l_p^3 m_1 m_p^4 x_2^2) \right\}$$

The Linearized Model

Since we will design a controller using methods for linear systems, we need to linearize the nonlinear state space model of the system around the origin.

$$x = (0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0), \quad u = (0 \ 0) \Rightarrow \dot{x}[t] \approx \mathcal{A} \cdot x[t] + \mathcal{B} \cdot u[t]$$

This can easily be obtained using the **Linearize** function in *Control System Professional*.

```
ss = Linearize[f, {x1, θ, x2, α},
  {{x1, 0}, {θ, 0}, {x2, 0},
  {α, 0}, {dx1, 0}, {dθ, 0}, {dx2, 0}, {dα, 0}},
  {{F1, 0}, {F2, 0}}] // Simplify;
Map[MatrixForm, ss]
```

$$\text{StateSpace} \left[\begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ -\frac{g h m_1}{J} & g - \frac{c g h m_s}{J} & -\frac{g h (m_2 + m_p)}{J} & 0 & 0 & 0 & 0 & 0 \\ \frac{g m_1}{J} & \frac{c g m_s}{J} & \frac{g (m_2 + m_p)}{J} & 0 & 0 & 0 & 0 & 0 \\ -\frac{g h m_1}{J} & g - \frac{c g h m_s}{J} & -\frac{g h (m_2 + m_p)}{J} & -\frac{g m_p}{m_2} & 0 & 0 & 0 & 0 \\ -\frac{g m_1}{J} & -\frac{c g m_s}{J} & -\frac{g (m_2 + m_p)}{J} & \frac{g (m_2 + m_p)}{l_p m_2} & 0 & 0 & 0 & 0 \end{pmatrix}, \right.$$

$$\begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{h^2}{J} + \frac{1}{m_1} & \frac{h^2}{J} \\ -\frac{h}{J} & -\frac{h}{J} \\ \frac{h^2}{J} & \frac{h^2}{J} + \frac{1}{m_2} \\ \frac{h}{J} & \frac{h}{J} - \frac{1}{l_p m_2} \end{pmatrix}, \left(\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \right)$$

We extract the matrices from the linearization and substitute with the physical values of the parameters to get numerical entities.

```
{A, B, C, D} = ss /. StateSpace → List /. physicalvalues;
```

```
A // MatrixForm
```

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ -0.606381 & 9.53078 & -0.479464 & 0 & 0 & 0 & 0 & 0 \\ 5.27288 & 2.42797 & 4.16925 & 0 & 0 & 0 & 0 & 0 \\ -0.606381 & 9.53078 & -0.479464 & -4.0875 & 0 & 0 & 0 & 0 \\ -5.27288 & -2.42797 & -4.16925 & 41.4851 & 0 & 0 & 0 & 0 \end{pmatrix}$$

```
B // MatrixForm
```

$$\begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1.17106 & 0.00826563 \\ -0.071875 & -0.071875 \\ 0.00826563 & 2.0916 \\ 0.071875 & -6.14703 \end{pmatrix}$$

```
C // MatrixForm
```

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

```
D // MatrixForm
```

$$\begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}$$

The linear model of the system allows us to use standard methods for controller design.

□ Controller Design

The method for computing the feedback law requires the linear model to be controllable. This implies that the system can be stabilized and that the method will find a numerical instantiation of $\mathcal{L}$ such that this is obtained. To check controllability we use the **Controllable** command in *Control System Professional*.

```
Controllable[ss /. physicalvalues]
```

```
True
```

Compute a full state feedback controller using the LQG-method (**LQRegulator**: **Gains** is a *Control System Professional* command).

```

L = LQRegulatorGains[
  ss /. physicalvalues,
  DiagonalMatrix[{1, 5., 1, 5., 0, 0, 0, 0}],
  0.1 IdentityMatrix[2] ];
L // MatrixForm

```

$$\begin{pmatrix} 26.7929 & 52.94 & 21.3676 & 11.8261 & 7.23982 & 18.3075 & 6.54028 & 2.2892 \\ -7.65183 & -26.7908 & -10.0775 & -27.7259 & -1.62889 & -8.77817 & -5.38841 & -4.54939 \end{pmatrix}$$

The control law to be used is $u[t] = -\mathcal{L} \cdot x[t]$, where $x[t]$ are measurements of the states. This gives the following closed loop system to simulate $\dot{x}[t] = f[x[t], -\mathcal{L} \cdot x[t]]$.

□ Simulation and Code Generation

Preliminaries

We store the states as a vector.

```

xss := {x1, θ, x2, α, dx1, dθ, dx2, dα}

```

We close the feedback loop, which gives the following right-hand side of the state-space equations.

```

f1 = f /. Thread[{F1, F2} → -L.xss];

```

Change the names of variables to only ASCII characters (needed for code generation).

```

Cvariables =
  {x1 → x1, θ → theta, x2 → x2, α → alpha,
   dx1 → dx1, dθ → dtheta, dx2 → dx2, dα → dalpha};

```

The right-hand side of the closed-loop state-space equations (intended for code generation).

```

rhs = f1 /. physicalvalues /. Cvariables;

```

We define the function **ClosedLoopPendulumEquationsRHS**, which has the huge right-hand side expression from above as its body. The type declarations **Real** and $\rightarrow \mathbf{Real}[8]$ below are the syntax in *MathCode C++* for providing type information. Explicit type declarations are needed to generate efficient C++ code.

```

ClosedLoopPendulumEquationsRHS[
  Real x1_, Real theta_, Real x2_, Real alpha_,
  Real dx1_, Real dtheta_, Real dx2_, Real dalpha_] → Real[8] =
  rhs;

```

Simulation of the Nonlinear Model within Mathematica

We define $f_{cl}[t]$ to be a short notation for the **ClosedLoopPendulumEquationsRHS**.

```

fcl[t_] := ClosedLoopPendulumEquationsRHS[x1[t], theta[t],
  x2[t], alpha[t], dx1[t], dtheta[t], dx2[t], dalpha[t]]

```

The state vector is defined.

```

x[t_] := {x1[t], theta[t], x2[t],
  alpha[t], dx1[t], dtheta[t], dx2[t], dalpha[t]};

```

The differential equations for the closed-loop system is given.

```
deq := Thread[x'[t] == fcl[t]]
```

We need to specify some initial conditions for the simulation. Assume that cart 1 is positioned at -1 , the seesaw is horizontal, cart 2 is positioned at $+1$, and the pendulum has a small deviation of 5.7° from the vertical axis. Furthermore, assume that the system is at rest at these positions (all time derivatives are zero).

```
initeq := Thread[x[0] == {-1, 0, 1, 0.1, 0, 0, 0, 0}]
```

Nonlinear System Simulation

We simulate the system using `NDSolve`.

```
dsolStandard = NDSolve[deq ∪ initeq, {x1, theta, x2,
    alpha, dx1, dtheta, dx2, dalpha}, {t, 0, 10}]; // Timing
NDSolveStandardTime = First[%];
{0.952Second, Null}
```

Linear System Simulation

```
dlineq := Thread[x'[t] == (A - B.L).x[t]] // Chop
```

We simulate the linear system using `NDSolve`.

```
dsollin = NDSolve[dlineq ∪ initeq,
    {x1, theta, x2, alpha, dx1, dtheta, dx2, dalpha}, {t, 0, 8}];
```

Plot the result for the first four state variables from the nonlinear simulation together with the result for the linear system.

```
Map[Plot[#, {t, 0, 8}, PlotRange → All, DisplayFunction → Identity] &,
    {Part[Flatten[x[t] /. dsolStandard], Range[4]],
    Part[Flatten[x[t] /. dsollin], Range[4]]} // Transpose];
Show[GraphicsArray[Partition[%, 2]], Display
    Function → $DisplayFunction];
```

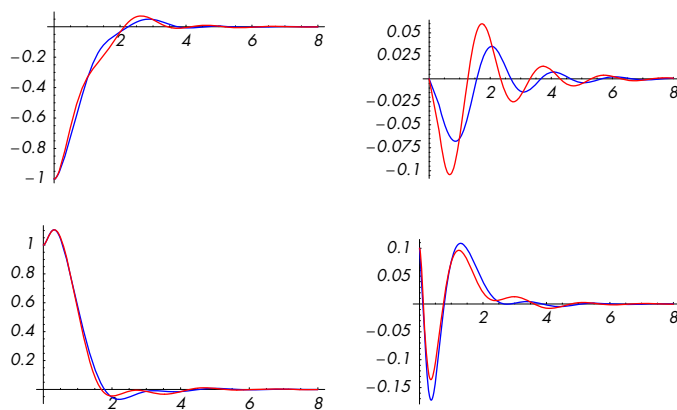


Figure 2. Initial value response for the linear (red) and nonlinear (blue) systems.

We notice that the linear response differs significantly from the nonlinear one. The difference between the linear and nonlinear response will be much smaller if the initial values are chosen to be smaller since the linear model only is a good approximation for small values of the states and inputs.

External Simulation of the Nonlinear Model

To generate simulation code to be run outside *Mathematica*, we have to provide a differential equation solver that can be compiled or linked when the executable program is generated. In this case, we have chosen to implement a very simple Runge–Kutta solver in *Mathematica*, which will be included in the generated code. Another solution could be to use a solver from a publicly available software library like the solvers at www.netlib.org.

Code for External Simulation of the Nonlinear Model

Here follows an implementation of a state equation solver using the Runge–Kutta method.

```

RK[Integer n_, Real h_, Real t0_, Real[_] startv_,
  Integer dimen_, Integer dimPlusOne_] → Real[n, dimPlusOne] :=
Module[{
  Real[dimen] {x, k1, k2, k3, k4},
  Integer i,
  Real t,
  Real[n, dimPlusOne] res
},
  t = t0;
  x = startv;
  res[[1, 1]] = t;
  res[[1, 2 | dimPlusOne]] = x;
  For[i = 1, i ≤ n - 1, i++,
    k1 = h ClosedLoopPendulumEquationsRHS[
      x[[1]], x[[2]], x[[3]], x[[4]], x[[5]], x[[6]], x[[7]], x[[8]]];
    k2 = h ClosedLoopPendulumEquationsRHS[x[[1]] + k1[[1]]/2,
      x[[2]] + k1[[2]]/2, x[[3]] + k1[[3]]/2, x[[4]] + k1[[4]]/2, x[[5]] + k1[[5]]/2,
      x[[6]] + k1[[6]]/2, x[[7]] + k1[[7]]/2, x[[8]] + k1[[8]]/2];
    k3 = h ClosedLoopPendulumEquationsRHS[x[[1]] + k2[[1]]/2,
      x[[2]] + k2[[2]]/2, x[[3]] + k2[[3]]/2, x[[4]] + k2[[4]]/2, x[[5]] + k2[[5]]/2,
      x[[6]] + k2[[6]]/2, x[[7]] + k2[[7]]/2, x[[8]] + k2[[8]]/2];
    k4 = h ClosedLoopPendulumEquationsRHS[
      x[[1]] + k3[[1]],
      x[[2]] + k3[[2]], x[[3]] + k3[[3]], x[[4]] + k3[[4]], x[[5]] + k3[[5]],
      x[[6]] + k3[[6]], x[[7]] + k3[[7]], x[[8]] + k3[[8]]];
    x = x + 1/6 (k1 + 2 k2 + 2 k3 + k4);
    t = t + h;
    res[[i + 1, 1]] = t;
    res[[i + 1, 2 | dimPlusOne]] = x;
  ];
  res
]

```

We use **RK** in the following ODE solver.

```

ndSolve[initvalue_, maxstepsize_, {x_, xmin_, xmax_}] :=
Module[{n, dimen},
  n = IntegerPart[(xmax - xmin) / maxstepsize] + 1;
  dimen = Length[initvalue];
  res = RK[n, maxstepsize, xmin, initvalue, dimen, dimen + 1];
  Array[(Interpolation[AppendRows[
    Transpose[{res[[_, 1]]}], Transpose[{res[[_, # + 1]]}]]]) &, {8}]
]

```

To get a correct estimate of the time spent by the external code for simulating the system, we define the following functions that repeat the external simulation n times.

```

RKRepeated[Integer n_, Real h_, Real t0_,
  Real[_] startv_, Integer dimen_, Integer dimPlusOne_,
  Integer loops_] → Real[n, dimPlusOne] :=
Module[{Real[n, dimPlusOne] res},
  Do[res = RK[n, h, t0, startv, dimen, dimPlusOne], {loops}];
  res
]

ndSolveRepeated[
  initvalue_, maxstepsize_, {x_, xmin_, xmax_}, loops_] :=
Module[{n, dimen},
  n = IntegerPart[(xmax - xmin) / maxstepsize] + 1;
  dimen = Length[initvalue];
  res = RKRepeated[n, maxstepsize,
    xmin, initvalue, dimen, dimen + 1, loops];
  Array[(Interpolation[AppendRows[Transpose[{res[[_, 1]]}],
    Transpose[{res[[_, # + 1]]}]]]) &, {8}]
]

```

Compilation

We are now ready to compile the package. The right-hand side of the state equations are optimized using common subexpression elimination.

```
CompilePackage[EvaluateFunctions → {ClosedLoopPendulumEquationsRHS}]
```

Successful compilation to C++: 3 function(s)

The corresponding code can be inspected in a text editor.

```
Run["Notepad Global.cc"];
```

All necessary files for building binaries are now generated. The **MakeBinary** command can build either a standalone application using the code specified by an option **GenerateMainFileAndFunction** or a *MathLink* version, which can be easily installed into *Mathematica*. We build the *MathLink* version.

```
MakeBinary[];
```

To use the generated code for the simulation, we only have to install the code using the **InstallCode** command from *MathCode C++*.

External Simulation: Uncompiled Code (Computations within *Mathematica*)

```

dsolUncompiled =
  ndSolve[{-1, 0, 1, 0.1, 0, 0, 0, 0}, 0.01, {t, 0., 10.}]; // Timing
ndSolveUncompiledTime = First[%];

{85.142Second, Null}

```

We plot the result to verify the Runge–Kutta solver.

```

Map[Plot[#t], {t, 0, 8}, PlotRange → All, DisplayFunction → Identity,
  PlotStyle → Blue] &, Part[dsolUncompiled, Range[4]]];

Show[GraphicsArray[Partition[%, 2]],
  DisplayFunction → $DisplayFunction];

```

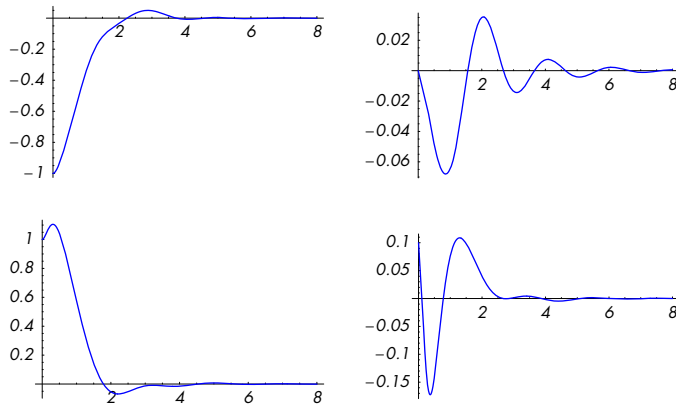


Figure 3. Initial value response for the nonlinear system computed internally by *Mathematica*.

External Simulation: Compiled Code (External Computations)

We install the executable code.

```
InstallCode[];
```

Global is installed.

The external simulation is performed 200 times to get an accurate measure of its timing.

```

dsolCompiled = ndSolveRepeated[
  {-1, 0, 1, 0.1, 0, 0, 0, 0}, 0.01, {t, 0., 10.}, 200]; // AbsTime
ndSolveCompiledTime = First[%] / 200;

{12.00000Second, Null}

```

We plot the result to verify the compiled Runge–Kutta solver.

```

Map[Plot[#t], {t, 0, 8}, PlotRange → All, DisplayFunction → Identity,
  PlotStyle → Blue] &, Part[dsolCompiled, Range[4]]];

```

```
Show[GraphicsArray[Partition[%, 2]],
      DisplayFunction -> $DisplayFunction];
```

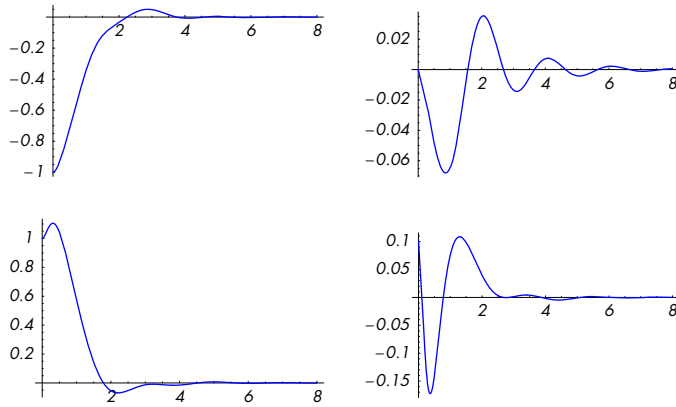


Figure 4. Initial value response for the nonlinear system computed by the external code.

The responses in Figure 3 and Figure 4 are identical.

Performance Comparisons

The difference in performance between the uncompiled and compiled version of the ODE solver is 1419.03.

```
ndSolveUncompiledTime / ndSolveCompiledTime
1419.03
```

A somewhat unfair comparison between the timings of the built-in ODE solver and the compiled solver gives 15.8667.

```
NDSolveStandardTime / ndSolveCompiledTime
15.8667
```

□ Clean-up

Uninstall the code and delete the temporary files.

```
UninstallCode[];
CleanMathCodeFiles[Confirm -> False, CleanAllBut -> {}];
```

Remove the functions for which we have generated code.

```
Remove[ClosedLoopPendulumEquationsRHS,
       RK, ndSolve, RKRepeated, ndSolveRepeated]
```

Unset the symbol f storing the right-hand side of the state-space equations.

```
f = .;
```

■ 3. A Fighter Aircraft

In this section a controller for a linear system is designed using LQG techniques; see, for example, [1, 3]. The resulting filter is transformed into discrete form and C++ code is generated corresponding to the filter equations. An extremely simple simulation is used to compare the generated code with the uncompiled *Mathematica* functions.

□ The System

We will consider a linearization of a nonlinear aircraft dynamics for some specific flight conditions. We study only horizontal motions.

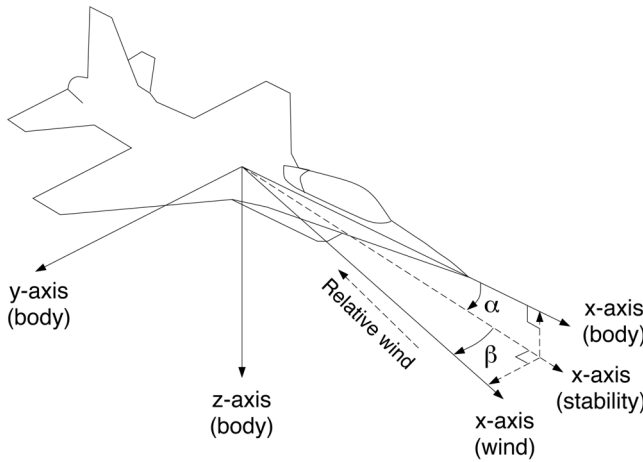


Figure 5. A fighter aircraft.

□ Modeling

Introduce the following notations:

- ψ course angle
- v_y velocity across the aircraft in the y -direction
- ϕ roll angle
- δ_a aileron deflection
- δ_r rudder deflection

and choose the state vector $x = \{v_y, p, r, \phi, \psi, \delta_a, \delta_r\}$, where $p \approx \dot{\phi}$ and $r \approx \dot{\psi}$.

Notation for a Linear Time Invariant (LTI) system is given in state-space form:

$$\begin{aligned}\dot{x} &= \mathcal{A}x + \mathcal{B}u + \mathcal{N}w \\ z &= \mathcal{M}x + \mathcal{D}_z u \\ y &= \mathcal{C}x + \mathcal{D}_y u + v\end{aligned}\tag{7}$$

The following numerical instantiations of the system matrices comes from an example in [1].

$$A = \begin{pmatrix} -0.292 & 8.13 & -202 & 9.77 & 0 & -12.5 & 17.1 \\ -0.152 & -2.54 & 0.561 & -0.0004 & 0 & 107 & 7.68 \\ 0.0364 & -0.0678 & -0.481 & 0.0012 & 0 & 4.67 & -7.98 \\ 0 & 1 & 0.0401 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -20 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -20 \end{pmatrix};$$

$$B = \begin{pmatrix} 0 & -2.15 \\ -31.7 & 0.0274 \\ 0 & 1.48 \\ 0 & 0 \\ 0 & 0 \\ 20 & 0 \\ 0 & 20 \end{pmatrix};$$

$$N = \begin{pmatrix} 0.292 & 0.001 & 0.97 & 0.0032 \\ 0.152 & -2.54 & 0.552 & 0.000043 \\ -0.0364 & -0.0688 & -0.456 & -0.00012 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix};$$

$$M = C = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix};$$

$$D_z = D_y = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix};$$

□ Controller Design

Suppose that we have the following wind disturbance acting on the system.

$$\begin{aligned} w_1[t_] &:= 10\,t \, ; \, 0 \leq t \wedge t \leq 1 \\ w_1[t_] &:= 10 \, ; \, 1 < t \wedge t < 2 \\ w_1[t_] &:= -10\,(t-3) \, ; \, 2 \leq t \wedge t \leq 3 \\ w_1[t_] &:= 0 \, ; \, t > 3 \end{aligned}$$

$$\begin{aligned} w_4[t_] &:= 10 \, ; \, 0 \leq t \wedge t \leq 1 \\ w_4[t_] &:= 0 \, ; \, 1 < t \wedge t < 2 \\ w_4[t_] &:= -10 \, ; \, 2 \leq t \wedge t \leq 3 \\ w_4[t_] &:= 0 \, ; \, t > 3 \end{aligned}$$

```
Plot[{w1[t], w4[t]}, {t, 0, 10},
  PlotLabel → "A wind disturbance", AxesLabel → {"t", "w1, w4"}];
```

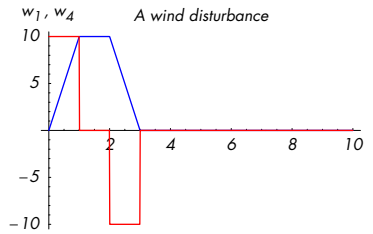


Figure 6. Some wind disturbances acting on the aircraft.

How will the system respond to these disturbances? We plot a simulation of the system response for 10 seconds.

```
plot1 = SimulationPlot[StateSpace[A, N, M, 02×4], {w1[t], 0, 0, w4[t]},
  {t, 0, 10}, Sampled → Period[0.05], PlotRange → All,
  PlotStyle → {Blue, Red}, AxesLabel → {"t", "φ, ψ"}];
```

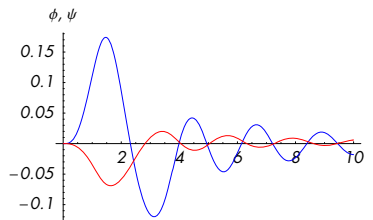


Figure 7. The response in roll angle ϕ (blue) and course angle ψ (red) to the wind disturbances plotted in Figure 6.

An LQ State-Feedback Design

We compute a state-feedback, that is, an $\mathcal{L}$ matrix, by minimizing a quadratic criterion with weight matrices Q_1 and Q_2 . The quadratic criterion is given by

$$\min_{\mathcal{L}} \int_0^{\infty} x^T[t] Q_1 x[t] + u^T[t] Q_2 u[t] dt \quad (8)$$

We play around with the matrices Q_1 and Q_2 until we are satisfied with the result.

```
Q1 = I2;
```

```
Q2 = 0.1 I2;
```

```
L = LQOutputRegulatorGains[StateSpace[A, B, M, Dz], Q1, Q2]
```

```
{{-0.00425964, 0.442824, 0.155747, 3.14624,
  0.369764, 1.69149, 0.0369027}, {-0.00789519, 0.0587867,
  -0.836838, 0.305382, -3.14059, 0.142669, 0.26782}}
```

The closed-loop eigenvalues are given.

A - B.L // Eigenvalues // TableForm

```
-21.8913
-20.0173
-9.49576 + 11.3361 i
-9.49576 - 11.3361 i
-2.35171 + 3.73879 i
-2.35171 - 3.73879 i
-0.178267
```

A necessary and sufficient condition for asymptotic stability of the closed-loop system is that these eigenvalues belong to the left half-plane.

We simulate the closed-loop output response.

```
plot2 =
SimulationPlot[StateSpace[A - B.L, N, M, 02x4], {w1[t], 0, 0, w4[t]},
{t, 0, 10}, Sampled → Period[0.05], PlotRange → All, PlotStyle →
{{Blue, Dashing[{.02, .02}]}, {Red, Dashing[{.02, .02}]}}];
```

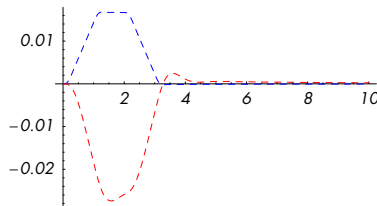


Figure 8. The response in roll angle ϕ (blue) and course angle ψ (red) to the wind disturbances plotted in Figure 6 for the controlled system.

We compare the disturbance attenuation with (dashed) and without the controller.

```
Show[plot1, plot2];
```

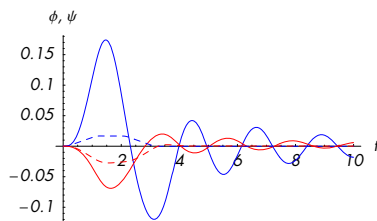


Figure 9. A comparison between the response in roll angle ϕ (blue) and course angle ψ (red) to the wind disturbances plotted in Figure 6 for the controlled (dashed) and uncontrolled system, respectively.

Kalman Filter

Now suppose that the states cannot be measured. Here we will design a linear dynamic controller known as a Kalman filter, which is used to estimate the states from measured signals.

Assume that the noise has the following covariance matrices.

$$\mathbf{R}_1 = 10^{-2} \mathbf{I}_4;$$

$$\mathbf{R}_2 = 10^{-6} \mathbf{I}_2;$$

$$\mathbf{K} = \text{LQEstimatorGains}[\text{StateSpace}[\mathbf{A}, \mathbf{N}, \mathbf{C}], \mathbf{R}_1, \mathbf{R}_2]$$

RiccatiSolve::meig :
Matrix with multiple eigenvalues encountered. Result may be inaccurate.

```
{-55.172, -1720.19}, {237.793, 15.4191}, {-1.66676, 34.8232},  
{21.7991, 0.502792}, {0.502792, 8.33028}, {0., 0.}, {0., 0.}
```

We check the eigenvalues of the dynamic controller to verify that it is stable.

```
 $\mathbf{A} - \mathbf{K} \cdot \mathbf{C}$  // Eigenvalues // TableForm
```

```
-20.  
-20.  
-11.4064 + 11.4047 i  
-11.4064 - 11.4047 i  
-4.48281 + 5.19271 i  
-4.48281 - 5.19271 i  
-0.203896
```

A Dynamic Controller

The Kalman filter equations are

$$\dot{x}_e = \mathcal{A}x_e + \mathcal{B}u + K(y - Cx_e) \quad (9)$$

The computed state-feedback control law is of the form

$$u = -Lx + L_r r \quad (10)$$

if we add a reference signal r .

Using the separation principle [1, 3], we can use the estimated states in equation (9) instead of the true states and still get an optimal design. In that case, equations (9) and (10) give the following LQG controller.

$$\begin{aligned} \dot{x}_e &= (\mathcal{A} - \mathcal{B}L - KC)x_e + \mathcal{B}L_r r + Ky \\ u &= -Lx_e + L_r r \end{aligned} \quad (11)$$

Here we choose L_r such that the static gain from r to z becomes an identity matrix.

$$\mathbf{L}_r = (\mathcal{M} \cdot (\mathcal{B} \cdot \mathbf{L} - \mathcal{A})^{-1} \cdot \mathcal{B})^{-1}; \text{MatrixForm}[\mathbf{L}_r]$$

$$\begin{pmatrix} 3.15189 & 0.369764 \\ 0.371095 & -3.14059 \end{pmatrix}$$

We represent the controller given by equation (11) in state-space form.

```
contKalmanFilter = StateSpace[A - B.L - K.C,
  BlockMatrix[{B.L, K}], -L, BlockMatrix[{L, 0_{2 \times 2}}]];

```

The continuous Kalman filter above is transformed to discrete time for implementation purposes.

```
discKalmanFilter =
  Chop[ToDiscreteTime[contKalmanFilter, Sampled -> Period[0.1]], 10^-8];

```

We define a simple test signal, which corresponds to a unit step in the reference signal for the roll angle ϕ while keeping the reference signal for the course angle ψ and the sensors for measuring the output signals ϕ and ψ equal to zero.

```
ry = Table[{1, 0, 0, 0}, {10}]^T;
SimulationPlot[discKalmanFilter,
  ry, PlotRange -> All, PlotStyle -> {Blue, Red}];

```

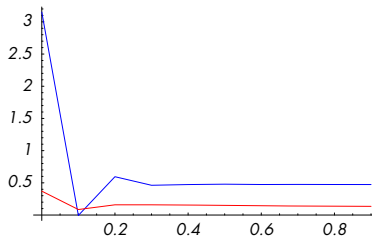


Figure 10. The controller outputs u_1 and u_2 when sensor signals are kept to zero and there is a unit step in the reference signal for the roll angle ϕ .

□ Simulation and Code Generation

The system matrices of the discrete Kalman filter are declared as global real constants (type information given to *MathCode C++*).

```
Declare[
  Constant Real[_] Ac = discKalmanFilter[[1]],
  Constant Real[_] Bc = discKalmanFilter[[2]],
  Constant Real[_] Cc = discKalmanFilter[[3]],
  Constant Real Dc = discKalmanFilter[[4]]
]

```

Define the system functions (right-hand sides of the controller equations).

```
f[Real[7] xe_, Real[4] in_] -> Real[_] := Ac.xe + Bc.in;
g[Real[7] xe_, Real[4] in_] -> Real[_] := Cc.xe + Dc.in;

```

We perform a simulation of the controller equations using these right-hand sides with a specified initial condition.

```
x0 = {0, 0, 0, 0, 0, 0, 0};

```

FoldList can be used to compute an iteration, $x_{n+1} = f[x_n, u_n]$, that is, a discrete simulation.

```
xesim = FoldList[f, x0, ryT];
```

Remove the last value.

```
xesim = Drop[xesim, -1];
```

Compute the output from the calculated input and state.

```
usim = MapThread[g, {xesim, ryT}]
```

Plot the output u_2 with samples instead of time on the x axis.

```
ListPlot[(usimT)[[2]], PlotJoined → True, PlotStyle → Red];
```

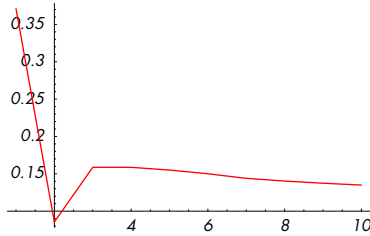


Figure 11. The controller output u_2 when sensor signals are kept to zero and there is a unit step in the reference signal for the roll angle ϕ .

Create a loop that will be the main loop in the external code.

```
DiscreteSimulate[Real[n, _] u_, Real[_] x0_] → Real[_] :=
Module[
{Real[_] x = x0,
Real[n, 2] y = 0,
Integer i},
For[i = 1, i ≤ n, i++,
y[[i]] = g[x, u[[i]]];
x = f[x, u[[i]]]
];
y
];
```

By construction, the functions **f** and **g** can be expanded to remove the dot-products and to reduce the computational effort. This is especially good when the system matrices contain many zeroes. The expansion is achieved using the **EvaluateFunctions** option.

Generate the code and compile it. The functions **f** and **g** will be expanded and subexpressions optimized before being code-generated.

```
BuildCode[EvaluateFunctions → {f, g}]
```

Successful compilation to C++: 3 function(s)

We inspect the resulting C++ code.

```
Run["Notepad Global.cc"];
```

Install the compiled code into *Mathematica* and test if the same simulation result as in Figure 11 is achieved.

```
InstallCode[];
```

```
Global is installed.
```

```
usim2 = DiscreteSimulate[ryT, x0];
```

Compare the controller outputs computed by *Mathematica* and by the external code. The maximum absolute error in the given sequence.

```
Max[Abs[usim - usim2]]
```

```
 $3.33067 \times 10^{-16}$ 
```

Plot the simulation when the compiled code is used.

```
ListPlot[(usim2T)[[2]], PlotJoined → True, PlotStyle → Red];
```

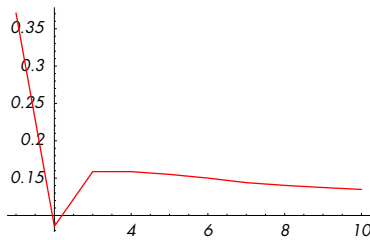


Figure 12. The same plot as in Figure 11 but computed by the external code.

We observe that the external simulation gives the same result as the simulation computed within *Mathematica*.

□ Clean-up

Uninstall the code and delete the temporary files.

```
UninstallCode[];
```

```
CleanMathCodeFiles[Confirm → False, CleanAllBut → {}];
```

Delete the temporary directory and all files it contains.

```
SetDirectory[ToFileName[".."]];
```

```
DeleteDirectory["Tmp", DeleteContents → True];
```

■ 4. Conclusions

In this notebook, we have demonstrated the use of *Mathematica* in modeling of dynamic systems and controller design. The symbolic capabilities of *Mathematica* are very useful for deriving dynamic equations according to the Lagrangian

formalism. Furthermore, rewriting these equations in state-space form, suitable for controller design, is also easily done using **Solve**. A unique solution is always obtained since the unknown second-order derivatives always appear linearly in the equations derived from the partial differential equation (6) that the Lagrangian of the system has to satisfy.

We have also shown how efficient C++ code for simulation can be generated using the application package *MathCode* C++. The pendulum/seesaw example is only a toy example compared to many of them. Industrial applications motivate the efforts to be able to automatically generate C++ code that can be compiled and run outside *Mathematica* for increased performance.

In the fighter aircraft example, we illustrated how *Mathematica* can be used as a prototyping environment for controller design. First the model of the system is analyzed and different controllers can be evaluated. When a satisfactory solution has been found, *MathCode* C++ can be used to generate standalone C++ code that can be used in the real application.

■ Initialization

In this section, we load a number of add-on packages, define auxiliary functions, and set up the environment for C++ code generation.

Load the *Control System Professional* package.

```
Needs["ControlSystems`Master`"];
```

Load the *MathCode* C++ package.

```
Needs["MathCode`"];
```

MathCode C++ ver 1.10 loaded.

Since a number of temporary files will be generated by *MathCode* C++, we create a temporary directory and set this to the current working directory.

```
CreateDirectory["Tmp"];
SetDirectory[ToFileName[{".", "Tmp"}]];
```

This function will be used to measure the time for external processes. We cannot rely on the **Timing** command since it includes only CPU time spent in the *Mathematica* kernel. It does not include time spent in external processes connected via *MathLink*.

```
SetAttributes[AbsTime, HoldFirst];
AbsTime[x_] :=
Module[{start, res},
  start = AbsoluteTime[];
  res = x;
  {(AbsoluteTime[] - start) Second, res}
];
```

Give the notation for colors.

```
Needs["Graphics`Colors`"]
```

Set the default color list for multiple plots.

```
SetOptions[Plot,  
  PlotStyle → {Blue, Red, Green, Cyan, Magenta, Yellow}];
```

Load the user-defined package that gives improved notation for matrix operations.

```
Needs["MatrixNotation`"]
```

Turn off the real-time spell checking.

```
Off[General::"spell1", General::"spell"];
```

■ References

- [1] T. Glad and L. Ljung, *Control Theory: Multivariable and Nonlinear Methods*, London; New York: Taylor & Francis, 2000.
- [2] L. Ljung and T. Glad, *Modeling of Dynamic Systems*, Englewood Cliffs, NJ: Prentice Hall, 1994.
- [3] J. M. Maciejowski, *Multivariable Feedback Design. Electronic Systems Engineering Series*, Wokingham, UK; Reading, MA: Addison-Wesley, 1989.
- [4] K. Zhou, J.C. Doyle, and K. Glover, *Robust and Optimal Control*, Upper Saddle River, NJ: Prentice Hall, 1996.
- [5] H. Goldstein, *Classical Mechanics*, 2nd ed., Reading, MA: Addison-Wesley, 1980.

■ Additional Material

Available at www.mathematica-journal.com.

MatrixNotation.nb
MatrixNotation.m

About the Authors

Mats Jirstrand is co-founder and VP of Applications for MathCore AB, developers of MathCode C++ and MathModelica. He received an M.S. in Applied Physics and Electrical Engineering from Linköping University, Sweden, in 1994 and a Ph.D. in Automatic Control also from Linköping University in 1998.

Johan Gunnarsson is co-founder and VP of Development for MathCore AB. He received an M.S. in Applied Physics and Electrical Engineering from Linköping University, Sweden, in 1993 and a Ph.D. in Automatic Control also from Linköping University in 1997.

Musical and Graphic Representations of the Fibonacci String and Proteins Using Mathematica

Erik Jensen

University of California, Santa Cruz

Ronald J. Rusay

Diablo Valley College, Pleasant Hill, California

and

University of California, Berkeley

■ Introduction

There are many types of natural phenomena that can be represented by strings of characters. For example, the primary sequence of a protein is the order of amino acids that composes the macromolecular string. There are about twenty amino acids which are used as the building blocks of proteins. Each amino acid has a unique set of chemical and structural properties. Consequently, proteins can be conceptualized as finite words over a twenty-letter alphabet. The Fibonacci sequence is an infinite series that is derived from a model invented by Leonardo Pisano in the early thirteenth century to describe the population growth of rabbits. The series follows the recursive function $F_n = F_{n-1} + F_{n-2}$. This

Originally presented at the Third International *Mathematica* Symposium (Hagenberg, Austria, August 23–25, 1999).

sequence has been observed in many other systems and geometries found in nature. For example, it has been observed that many species of flowers have a total number of petals that is equal to a Fibonacci number. The proportions of many natural spirals such as those of pine cones, snail shells and nautilus shells also follow the Fibonacci sequence. Associated with this sequence is a binary ω -word generated by the morphism $\{b(a) = a\,b, b(b) = a\}$.

Mathematica can be used to perform many operations on strings. As a result, it can be used to produce unique demonstrations of the order inherent in certain strings. The focus of this project has been the development of graphic and musical demonstrations which reflect the order and symmetry inherent in strings found in nature.

■ Music and Graphics of Proteins

The following program has been designed to associate a list of notes to amino acids in a protein using a logical ordering system in the scale of C Minor. Two parallel scales are employed. The polar amino acids are assigned notes of increasing pitch on the basis of increasing molecular complexity of their side chains. The same scale is assigned to the nonpolar amino acids using the same ordering system, but the notes are assigned a different duration and slightly different tonal quality. The polarity of a molecule is one characteristic used in biochemistry to classify the properties of amino acids. It would be possible to assign notes on the basis of other biochemical properties of amino acids as well. Since the chemical properties of the amino acids and their interactions determine the structure and function of a protein, an association between chemical characteristics and notes may result in music that reflects these attributes. The goal is to have the characteristics of the sounds correspond as much as possible to the characteristics of the amino acids so that their patterns within the molecule may be apparent in the musical combinations of their sounds.

```

In[1]:= protein = {g, a, v, a, i, t};
        polar = {g, a, v, l, i, m, f, w, p};
        nonpolar = {s, t, c, y, n, g, q, d, e, k, r, h};
        freqtable =
            {262, 294, 311, 349, 392, 415, 466, 524, 588, 622, 698, 784};

In[5]:= (* build rule tables *)
        polarrules =
            MapThread[(#1 → Play[Sin[2  $\pi$  #2 t] + Sin[2  $\pi$  (#2 + 1) t], {t, 0, .45},
                        DisplayFunction → Identity]) &,
                        {polar, Take[freqtable, Length[polar]]}];

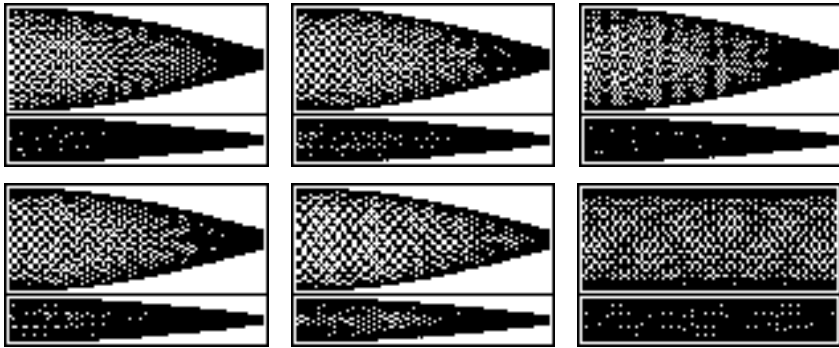
In[6]:= nonpolarrules = MapThread[
            #1 → Play[Sin[2  $\pi$  #2 t], {t, 0, .75}, DisplayFunction → Identity] &,
            {nonpolar, Take[freqtable, Length[nonpolar]]}];

In[7]:= song = (protein /. polarrules) /. nonpolarrules ;

```

```
In[8]:= Show /@ song;
```

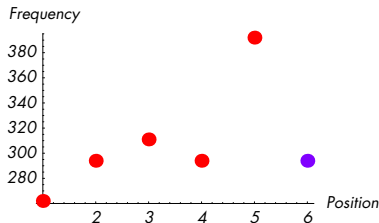
From In[8]:=



The sequence of notes produced by the previous program can also be plotted graphically. This program plots the notes played using the same ordering system and distinguishes between the polar and nonpolar amino acids.

```
In[9]:= polarrules2 = MapThread[({#1 → {#2, Hue[0]}}) &,
  {polar, Take[freqtable, Length[polar]]}];
In[10]:= nonpolarrules2 = MapThread[({#1 → {#2, Hue[.75]}}) &,
  {nonpolar, Take[freqtable, Length[nonpolar]]}];
In[11]:= song2 = (protein /. polarrules2) /. nonpolarrules2;
In[12]:= Show[Graphics[{{PointSize[.05],
  MapIndexed[{{#1[[2]], Point[{{#2[[1]], #[[1]]}}] &, song2}}, Axes → True,
  PlotRange → All, AxesLabel → {"Position", "Frequency"}]]];
```

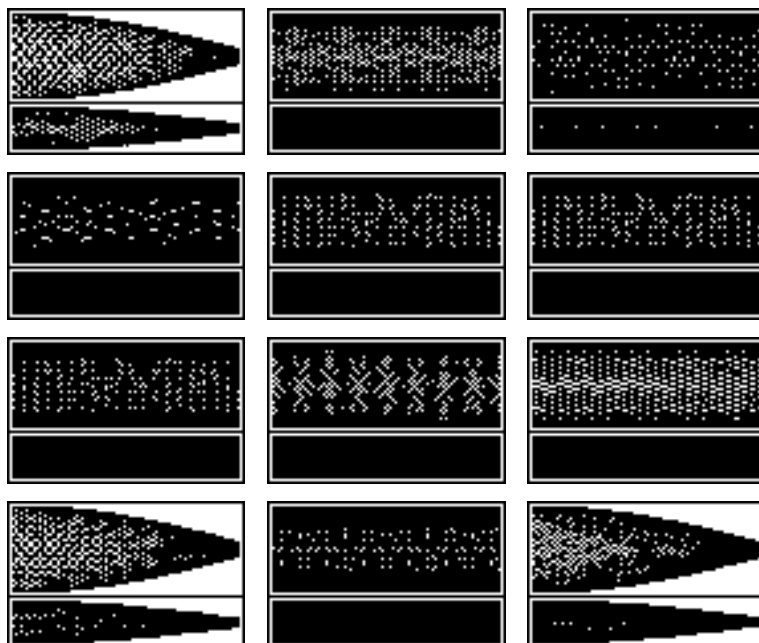
From In[12]:=



A pseudo-random list of amino acids can be created to be used as the input for the previous music and graphic programs. In this way, the primary sequence of proteins can be compared to pseudo-random lists of amino acids.

```
In[13]:= bases = {g, a, v, l, i, m, f, w, p, s, t, c, y, n, g, q, d, e, k, r, h};
  protein = Table[bases[Random[Integer, {1, Length[bases]}]], {12}];
In[15]:= song = (protein /. polarrules) /. nonpolarrules;
```

```
In[16]:= Show /@ song;
```

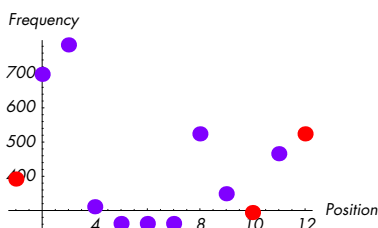


From In[16]:=

```
In[17]:= song2 = (protein /. polarrules2) /. nonpolarrules2;
```

```
In[18]:= Show[Graphics[{{PointSize[.05],
  MapIndexed[{{#1[[2]], Point[{{#2[[1]], #1[[1]]}}] &, song2}}, Axes -> True,
  PlotRange -> All, AxesLabel -> {"Position", "Frequency"}]]];
```

From In[18]:=



■ Music and Graphics of the Fibonacci String

The Fibonacci string is a binary ω -word generated by the morphism $\{b(a) = ab, b(b) = a\}$. It is aperiodic. One interesting property of the Fibonacci string is that substrings repeat with spaces between repetitions that follow the exact pattern of the Fibonacci string itself. That is, the intervals are equal to $\{a, b, a, a, b, \dots\}$ in succession where “ a ” and “ b ” are integers. The following

program demonstrates this property by generating the string, identifying the positioning of a specified substring, determining the spacing between them, and then converting the resulting integers to the symbols “a” and “b”. When the last step is performed, the Fibonacci string re-emerges.

```
In[19]:= stringpos = Flatten[
      StringPosition[StringJoin[Map[ToString, Flatten[ReplaceRepeated[
        0, {0 → 1, 1 → {1, 0}}, MaxIterations → 12]]], "1011"]];

In[20]:= distances = Subtract @@ Transpose[Partition[Rest[stringpos], 2]];

In[21]:= distances /. {distances[[1]] → a, distances[[2]] → b}

Out[21]:= {a, b, a, a, b, a, b, a, a, b, a, a, b, a, b, a,
  a, b, a, b, a, a, b, a, a, b, a, b, a, a, b, a, a,
  b, a, a, b, a, b, a, a, b, a, a, b, a, a, b, a, b}
```

Music can also be generated from the Fibonacci string. This next program creates music from the Fibonacci string by means of the following steps:

1. Generation of the string to a particular length;
2. Division of the string into substrings of length (n);
3. Generation of a list of all possible substrings of length (n);
4. Assignment of notes of ascending frequency in the scale of C Major to the previous list with a logical ordering system;
5. Transformation of the divided Fibonacci string according to the rules established in the last step.

This program produces a unique melody for each specified substring length (n).

```
In[22]:= len = 5;

In[23]:= strings = Partition[Flatten[
      ReplaceRepeated[0, {0 → 1, 1 → {1, 0}}, MaxIterations → 10]], len];

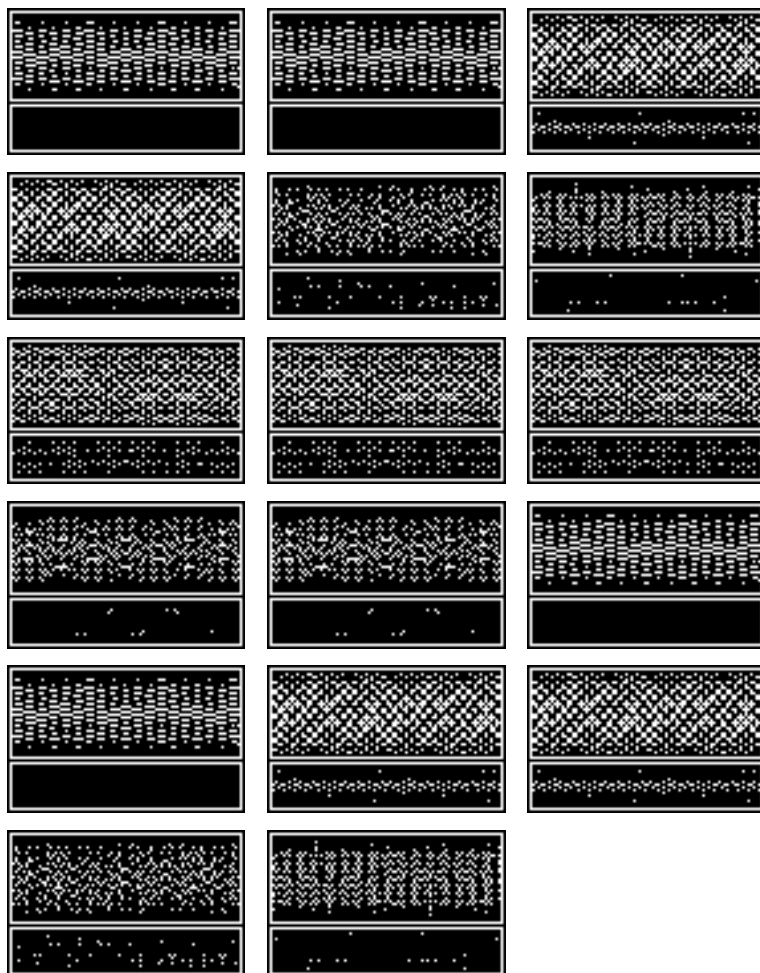
In[24]:= stable =
      Sort[
        Map[First,
          NestList[DeleteCases[#, First[#]] &, strings, len]]];

In[25]:= (* build rule tables *)
      rules = MapThread[
        (#1 → Play[Sin[2  $\pi$  #2 t], {t, 0, .5}, DisplayFunction → Identity]) &,
        {stable, Take[freqtable, Length[stable]]}];

In[26]:= song = strings /. rules;
```

In[27]:= Show /@ song;

From In[27]:=



The previous music can be represented graphically by the next program. It produces a unique graph for each specified length of substring (n). Of particular interest is the apparently regular oscillation between larger and smaller sets of values for any specified substring length. It functions by means of the following steps:

1. Generation of the string;
2. Division of string to specified substring length (n);
3. The substrings of the divided string are interpreted as numerical values and plotted with their specific position in the string to produce a graphic representation of the succession of substrings.

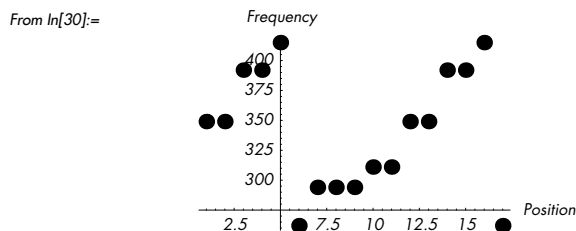
```

In[28]:= rules2 =
  MapThread[(#1 → #2) &, {stable, Take[freqtable, Length[stable]]}];

In[29]:= song2 = strings /. rules2;

In[30]:= Show[Graphics[{PointSize[.05],
  MapIndexed[{Point[{#2[[1]], #1}]} &, song2]}, Axes → True,
  PlotRange → All, AxesLabel → {"Position", "Frequency"}];

```



■ Spirals Based on the Fibonacci Sequence

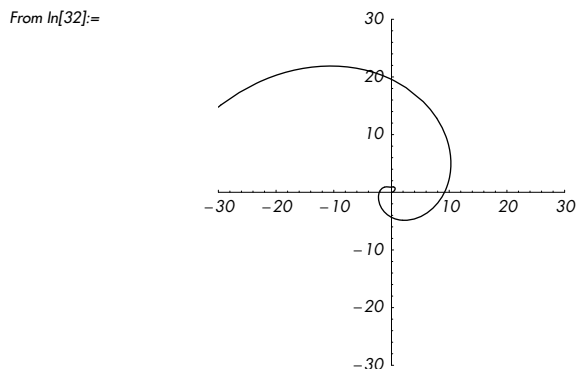
A polar coordinate system can be used to create spirals based on the Fibonacci sequence with *Mathematica*. The formulas used to determine the n th Fibonacci number yield complex results when n is a non-integer. The internal algorithm employed by the *Mathematica* command **Fibonacci**[n] drops the imaginary component of any complex results. As a result, $y = \text{Fibonacci}[x]$ is a continuous function and generally increasing in the set of positive real numbers. Therefore, $r = \text{Fibonacci}[\theta]$ yields a spiral. The next program uses a parametric coordinate system to generate such a spiral.

```

In[31]:= radius := Fibonacci[t];

In[32]:= ParametricPlot[radius {Cos[t], Sin[t]}, {t, 0, 10},
  PlotRange → {{-30, 30}, {-30, 30}}, AspectRatio → Automatic];

```



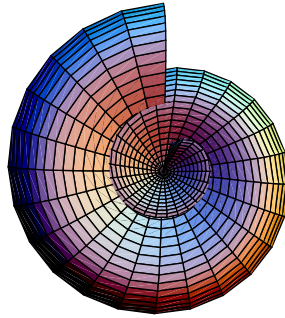
Another method of generating spirals from the Fibonacci sequence is based on the arrangement of adjacent squares that increase in size following the sequence of Fibonacci numbers. This method always yields logarithmic spirals. A line

drawn from the origin to any point on such a spiral will form a constant angle with the tangent line at that point. It is unknown whether the spirals $r = \text{Fibonacci}[\theta]$ or spirals produced in a similar manner possess any of the properties of the logarithmic spirals.

By manipulating the equations for the spirals in a three-dimensional parametric coordinate system, it is possible to produce graphic images that resemble shells.

```
In[33]:= radius := Fibonacci[Sqrt[t]];
In[34]:= ParametricPlot3D[radius {2 u Sin[t], -u Sin[u], 2 u Cos[t]}, {t, 0, 4 Pi},
    {u, 0, Pi}, AspectRatio -> Automatic, PlotPoints -> {50, 20},
    ViewPoint -> {0, -1, 0}, Boxed -> False, Axes -> False];
```

From In[34]:=



■ Further Possibilities

- Music from the Fibonacci string based on substrings of different lengths combined melodically into one song. The resultant music could reveal different characteristics of order within the string.
- Music from:
 - Strings that occur in mathematical problems (e.g., Tower of Hanoi and Lucas Number).
 - Strings that represent socioeconomic behavior such as stock market indices and consumer prices.
- Combining related strings into a single composition:

For example, the amino acid sequence of a protein and its corresponding DNA sequence may be played simultaneously. A harmonious synchronization of notes between the two strings could be produced where the DNA sequence serves as a rhythmic percussion sequence.
- *Mathematica* could be used to produce graphs of the spirals and tangent lines to the spirals simultaneously. The angles formed with a line from the origin could be measured to determine if these spirals are logarithmic or not.

■ References

- R. Knott, "The Fibonacci Rabbit sequence." (Sep 27, 2000),
<http://www.ee.surrey.ac.uk/Personal/R.Knott/Fibonacci/fibrab.html>.
- R. Knott, "Fibonacci Numbers and Nature." (Mar 31, 2001),
<http://www.ee.surrey.ac.uk/Personal/R.Knott/Fibonacci/fibnat.html>.
- R. Knott, "A Formula for Fib(n)." (Feb 13, 2001),
<http://www.ee.surrey.ac.uk/Personal/R.Knott/Fibonacci/fibFormula.html>.
- R. Rusay, V. Keränen, T. Laakso, E. Jensen, T. Dunham, "Music from DNA/Proteins/Math.",
<http://ep.llnl.gov/msds/music/midi-dna.html>.
- D. Schweizer, "The Fibonacci Numbers." (Nov 1, 1996),
<http://math.holycross.edu/~davids/fibonacci/fibonacci.html>.
- I. Galkin, "Fibonacci Numbers Spelled Out." (Dec 4, 1997),
<http://ulcar.uml.edu/~iag/CS/Fibonacci.html>.
- R. Knott, D.A. Quinney, and PASS Maths, "The Life and Numbers of Fibonacci," *Plus* [online],
3, 1997, <http://pass.maths.org.uk/issue3/fibonacci/index.html>.

About the Authors

Erik Jensen is currently completing his final year as an undergraduate at the University of California, Santa Cruz, where he studies biochemistry and molecular biology. Under the direction of Professor David Deamer, he is currently working on origins of life research, which includes developing a process that identifies catalytic RNA fragments within a solution of random RNA fragments. In addition, Erik independently pursues interests in cellular automata, complex systems, and bioinformatics.

Dr. Ron Rusay is a chemistry professor at Diablo Valley College, a visiting scholar at the University of California, Berkeley, and a Science Education Fellow at Lawrence Livermore National Laboratory. After completing Ph.D. studies in natural product chemistry and oceanography, he spent almost fifteen years in the chemical industry, where he held various research and management positions before returning to the classroom. A more detailed biography can be found at: ep.llnl.gov/msds/dvc/rusay.html

CFLP: A Mathematica Implementation of a Distributed Constraint Solving System

Mircea Marin, Tetsuo Ida

*Institute of Information Sciences and Electronics
University of Tsukuba
Tsukuba Ibaraki 305-8573, Japan
mmarin@score.is.tsukuba.ac.jp, ida@score.is.tsukuba.ac.jp*

Wolfgang Schreiner

*Research Institute for Symbolic Computation (RISC-Linz)
Johannes Kepler University
A-4040 Linz, Austria
Wolfgang.Schreiner@risc.uni-linz.ac.at*

The need for combining and making various constraint solvers cooperate is widely recognized. Such an integrated system would allow solving problems that cannot be solved by a single solver.

CFLP (Constraint Functional Logic Programming System) is a distributed software system consisting of a functional logic programming interpreter running on one machine and a number of constraint-solving engines running on other machines. The interpreter is based on a deterministic version of a lazy narrowing calculus which was extended in two main directions: (a) the possibility to specify explicit OR-parallelism, and (b) the possibility to specify constraints over various domains. The OR-parallel features of the interpreter allow the decomposition of the solution space into different subspaces denoted by various sets of constraints; the individ-

Originally presented at the Third International *Mathematica* Symposium (Hagenberg, Austria, August 23–25, 1999).

ual sets are solved by different constraint-solving engines in parallel and joined together to form the total solution set. This allows the user to investigate problems with large solution spaces using the computational power available in large computer networks.

The CFLP system is written entirely in *Mathematica* and uses the *Math-Link* protocol for interprocess communication. The current implementation can solve problems expressible in functional logic and involving constraints such as systems of linear, polynomial, and differential equations, and equations with invertible functions.

■ Introduction

Recent refinements of lazy narrowing calculi for solving equations over the domain of the terms lead to successful implementations that can be used as the starting point for developing a system based on equational reasoning. There are, however, many applications such as theorem proving and computational geometry where the expressive power of pure functional logic programming is not sufficient. These problems usually involve solving systems of constraint over various, like polynomial equations, differential equations, linear equations, and inequations. The CLP (Constraint Logic Programming) scheme proposed by Jaffar and Lassez [1] showed the possibility of integrating constraint solving capabilities over domains like reals, booleans, finite domains, and rational and infinite trees in logic programming languages. Notably, almost all of these languages have a sequential implementation. Our investigation showed that it is possible to refine a lazy narrowing calculus with inference rules for handling constraints, and to develop a model for solving the constraints in a distributed environment.

The system is called CFLP (Constraint Functional Logic Programming System) [2], and consists of a functional logic interpreter based on a higher-order lazy narrowing calculus and a number of constraint solvers that may run on different machines. The interpreter is implemented in *Mathematica*. The cooperation between interpreter and constraint solvers is coordinated by a constraint scheduler, which schedules the individual constraint solvers to handle the solving tasks generated by the interpreter.

The paper is structured as follows. First, we illustrate by example the system capabilities. Next, we describe the system architecture. The following subsections describe the three main components of the system: interpreter, scheduler, and constraint solvers. Finally, we draw some conclusions and directions of further research.

■ Examples

CFLP is delivered as a collection of *Mathematica* packages, which add solving power to a normal *Mathematica* session. If CFLP is properly installed, a *Mathematica* user can enable it by loading the package **TSolve.m**.

```
In[1]:= Needs["TSolve`"]
```

In this section we illustrate the functionality of CFLP with a few examples.

□ A Polynomial Approximation Problem

Consider the problem of finding the relationship between the coefficients of a univariate polynomial $f \in \mathbb{C}[x]$ of degree 3 and a univariate polynomial $g \in \mathbb{C}[x]$ of degree 4 related by the constraint that they have the same values for $x = 1, 2, 3, 4$.

In CFLP, the polynomial f of degree 3 can be specified by the λ -term $\lambda[\{x\}, a x^3 + b x^2 + c x + d]$ where a, b, c, d are constants, and the polynomial g of degree 4 can be specified by the λ -term $\lambda[\{x\}, m x^4 + n x^3 + p x^2 + q x + r]$ where m, n, p, q, r are variables whose values have to be computed from the conditions

$$f[1] = g[1], f[2] = g[2], f[3] = g[3], f[4] = g[4]$$

which can be specified in the compact form

$$\text{map}[f, \{1, 2, 3, 4\}] = \text{map}[g, \{1, 2, 3, 4\}] \quad (1)$$

where $\text{map}[f, \text{list}]$ applies f to each element of list . In CFLP, map can be specified via the following functional logic program.

```
In[2]:= Prog = {
    map[F, {}] -> {},
    map[F, [H | T1]] -> [F[H] | map[F, T1]];

```

The underlined symbols denote the free variables in the rewrite rules, and $[H | T1]$ denotes a list with head H and tail $T1$.

To solve equation (1), we call

```
In[3]:= TSolve[map[λ[{x}, a x^3 + b x^2 + c x + d], {1, 2, 3, 4}] ≐
    map[λ[{x}, m x^4 + n x^3 + p x^2 + q x + r], {1, 2, 3, 4}],
    {a, b, c, d}, {},
    DefinedSymbol -> {map : (Float -> Float) × TyList[Float] -> TyList[Float],
        f : Float -> Float},
    Rules -> {map[F, {}] -> {}, map[F, [H | T1]] -> [F[H] | map[F, T1]]},
    Constructor -> {m : Float, n : Float, p : Float, q : Float, r : Float}]

```

Type checking program ...

Type checking goal ...

Enabling distributed constraint solving subsystem ...

$$\begin{aligned}
 &\{a \rightarrow \\
 &\quad 25m + n + p + q + \frac{1}{3} \left(-18m - 3q - \frac{11r}{2} + \frac{11}{2} (-24m + r) \right) + \frac{1}{4} (8m - 4p - 6q - \\
 &\quad 7r + 7(-24m + r) + 2 \left(18m + 3q + \frac{11r}{2} - \frac{11}{2} (-24m + r) \right)), \\
 &b \rightarrow \frac{1}{4} (-8m + 4p + 6q + 7r - 7(-24m + r) - \\
 &\quad 2 \left(18m + 3q + \frac{11r}{2} - \frac{11}{2} (-24m + r) \right)), \\
 &c \rightarrow \frac{1}{3} \left(18m + 3q + \frac{11r}{2} - \frac{11}{2} (-24m + r) \right), \\
 &d \rightarrow -24m + r \}
 \end{aligned}$$

$$\begin{aligned}
 Out[3] = &\{ \{ a \rightarrow 25m + n + p + q + \frac{1}{3} \left(-18m - 3q - \frac{11r}{2} + \frac{11}{2} (-24m + r) \right) + \frac{1}{4} \\
 &\quad (8m - 4p - 6q - 7r + 7(-24m + r) + 2 \left(18m + 3q + \frac{11r}{2} - \frac{11}{2} (-24m + r) \right)), \\
 &b \rightarrow \frac{1}{4} (-8m + 4p + 6q + 7r - 7(-24m + r) - \\
 &\quad 2 \left(18m + 3q + \frac{11r}{2} - \frac{11}{2} (-24m + r) \right)), \\
 &c \rightarrow \frac{1}{3} \left(18m + 3q + \frac{11r}{2} - \frac{11}{2} (-24m + r) \right), d \rightarrow -24m + r \} \}
 \end{aligned}$$

The equational symbol $\doteq$ denotes *strict* equality, that is, the fact that the *values* of the two sides of the equation coincide. We identify the value of an expression with the result of rewriting it to an expression without defined symbols.

The syntax call of **TSolve** is similar to the syntax call of **Solve**. The first three arguments specify the goal to be solved, the main variables of the goal, and the eliminatable variables of the goal, respectively. In addition, the user of CFLP can control the behavior of the computation through various **TSolve** options. In this example, the following options are used.

DefinedSymbol	the list of type-annotated symbols defined by the user
Rules	the list of definitions for the user-defined symbols (the functional logic program)
Constructor	the list of type-annotated data constructors

The constant coefficients of f are given as data constructors, and the unknown coefficients of g are given as main variables.

A **TSolve** call may yield an infinite number of solutions. In this case it cannot produce an output with the list of all solutions but it prints them out incrementally, as soon as they are computed. **TSolve** calls can be interrupted via a palette which is provided for the user convenience.

In this example, CFLP returns the unique solution

$$\left\{ \begin{aligned} d &\rightarrow -24m + r, \\ c &\rightarrow 15m + 7n - \frac{7}{6}(60m + 6n) + 3p - \frac{3}{2}(50m + 12n - 2(60m + 6n) + 2p) + q, \\ b &\rightarrow \frac{1}{2}(50m + 12n - 2(60m + 6n) + 2p), \quad a \rightarrow \frac{1}{6}(60m + 6n) \end{aligned} \right\}$$

which is printed out as soon as it is computed, and collected in the output of the **TSolve** call.

Note the use of higher-order variables and λ -abstractions in the specification of the goal and functional logic program. CFLP is able to handle equations involving operators defined outside the functional logic program. Furthermore, the computed answer is parametric, since r is a variable.

□ Electrical Circuits

This example illustrates the expressive power of the functional logic programming style extended with higher-order constructs (function variables and λ -abstractions) and capabilities to solve differential equations, linear equations, and systems of polynomial equations. We consider the problem of modeling the behavior in time of electrical circuits built from serial and parallel connections of elementary components such as resistors, inductors, and capacitors.

First we introduce a new type constructor **ElComp** with associated data types **res**, **ind**, **cap**, **serial**, and **parallel** to specify the characteristics of the electrical circuits. Thus, a resistor with resistance $R = 2$ is specified as **res**[2], a serial connection of two resistors R_1 and R_2 is specified as **serial**[{**res**[R_1], **res**[R_2]}], and a parallel connection of a capacitor C and an inductor L is specified as **parallel**[{**cap**[C], **ind**[L]}]. The keyword **TyList** is the CFLP-type constructor for lists.

```
In[4]:= TypeConstructor[ElComp = res[Float] | ind[Float] | cap[Float] |
      serial[TyList[ElComp]] | parallel[TyList[ElComp]]];
```

Next, we define a function **spec** which describes the behavior in time t of an electrical component as a function of current $i[t]$ and voltage $v[t]$ in the circuit. The CFLP definition of **spec** is recursive, where the base case describes the behavior in time of the elementary circuits (resistor, inductor, capacitor), and the inductive case describes the behavior in time of serial and parallel connections of electrical components.

The CFLP program is given below. We do not explain the underlying electronic laws since it should be easy to read them off from the program.

```

In[5]:= Prog =
{spec[res[r], v : Float → Float, i : Float → Float] →
  True ← {λ[{t}, v[t]] ≈ λ[{t}, r i[t]]},
spec[ind[l], v : Float → Float, i : Float → Float] →
  True ← λ[{t}, v[t]] ≈ λ[{t}, l i'[t]],
spec[cap[c], v : Float → Float, i : Float → Float] →
  True ← λ[{t}, i[t]] ≈ λ[{t}, c v'[t]],
spec[serial[{}], λ[{t}, 0], i : Float → Float] → True,
spec[serial[[comp | T]],
  v : Float → Float, i : Float → Float] → True ←
{spec[comp, v1, i] ≈ True,
spec[serial[T], v2, i] ≈ True,
λ[{t}, v[t]] == λ[{t}, v1[t] + v2[t]]},
spec[parallel[{}], v, λ[{t}, 0]] → True,
spec[parallel[[comp | T]], v, i] → True ←
{spec[comp, v, i1] ≈ True,
spec[parallel[T], v, i2] ≈ True,
λ[{t}, i[t]] == λ[{t}, i1[t] + i2[t]]}};

```

Consider the problem of finding the behavior in time of the current in an RLC circuit, under the restriction that the voltage is constant in time and the current was initially set to 0.

In this case, the goal which we want to solve (in variable *i*) is

```

In[6]:= G = {i[0] == 0, spec[serial[{res[R0], ind[L0], cap[C0]}],
  λ[{t : Float}, V], λ[{t : Float}, i[t]]] ≐ True};

```

R0, **L0**, **C0**, and **V** are data constructors which denote arbitrary but fixed characteristic values for the resistor, inductor, capacitor, and voltage of the circuit. Now we can ask CFLP to solve the problem.

```

In[7]:= TSolve[G, {i : Float → Float}, {},
Rules → Prog,
Constructor → {R0 : Float, L0 : Float, C0 : Float, V : Float},
DefinedSymbol → {spec : ElComp × (Float → Float) × (Float → Float) → Bool}]

```

Type checking program ...

Type checking goal ...

$$\left\{ i \rightarrow \lambda \left[\{t \cdot 1\}, C_0 \left(- \frac{e^{\frac{(-\sqrt{C_0} R_0 - \sqrt{-4 L_0 + C_0 R_0^2}) t + 1}}{2 \sqrt{C_0} L_0}} \frac{(-\sqrt{C_0} R_0 + \sqrt{-4 L_0 + C_0 R_0^2}) C_{1_2}}{2 \sqrt{C_0} L_0} + \frac{e^{\frac{(-\sqrt{C_0} R_0 + \sqrt{-4 L_0 + C_0 R_0^2}) t + 1}}{2 \sqrt{C_0} L_0}} \frac{(-\sqrt{C_0} R_0 + \sqrt{-4 L_0 + C_0 R_0^2}) C_{1_2}}{2 \sqrt{C_0} L_0} \right) \right] \right\}$$

$$Out[7]= \left\{ \left\{ i \rightarrow \lambda \left[\{t \bullet 1\}, CO \left(- \frac{e^{\frac{(-\sqrt{CO} \cdot RO - \sqrt{-4 \cdot LO + CO \cdot RO^2}) \cdot t+1}}{2 \sqrt{CO} \cdot LO}} \left(-\sqrt{CO} \cdot RO + \sqrt{-4 \cdot LO + CO \cdot RO^2} \right) C1_2}{2 \sqrt{CO} \cdot LO} + \frac{e^{\frac{(-\sqrt{CO} \cdot RO + \sqrt{-4 \cdot LO + CO \cdot RO^2}) \cdot t+1}}{2 \sqrt{CO} \cdot LO}} \left(-\sqrt{CO} \cdot RO + \sqrt{-4 \cdot LO + CO \cdot RO^2} \right) C1_2}{2 \sqrt{CO} \cdot LO} \right) \right] \right\} \right\}$$

The system computes the parametric solution

$$\left\{ i \rightarrow \lambda t. e^{\frac{(-\sqrt{CO} \cdot RO - \sqrt{-4 \cdot LO + CO \cdot RO^2}) \cdot t}{2 \sqrt{CO} \cdot LO}} C1_1 - e^{\frac{(-\sqrt{CO} \cdot RO + \sqrt{-4 \cdot LO + CO \cdot RO^2}) \cdot t}{2 \sqrt{CO} \cdot LO}} C1_1 \right\}$$

which is represented in *Mathematica* by

$$\left\{ i \rightarrow \lambda \left[\{t\}, e^{\frac{(-\sqrt{CO} \cdot RO - \sqrt{-4 \cdot LO + CO \cdot RO^2}) \cdot t}{2 \sqrt{CO} \cdot LO}} C1_1 - e^{\frac{(-\sqrt{CO} \cdot RO + \sqrt{-4 \cdot LO + CO \cdot RO^2}) \cdot t}{2 \sqrt{CO} \cdot LO}} C1_1 \right] \right\}$$

Note the usage of type annotations in the goal, list of variables, and options of the **TSolve** call. CFLP has integrated a polymorphic type checker to verify the type consistency of the program and goal provided by the user.

□ A Problem Involving Solver Cooperation

Consider the following program

```
In[8]:= Needs["TSolve"]
```

```
In[9]:= Prog = {f[X : Compl] → g[Y : Compl] ⇐ (X + Y == 3 ∨ X2 - Y == 9)};
```

in complex variables **X**, **Y**, and the goal

```
In[10]:= G = {f[X] ≈ g[Y], g[Y2] ≈ g[Z2 - 1],
λ[{x : Compl}, H'[x]] == λ[{x : Compl}, Z xy], H'[1] == 4};
```

with indeterminates **X**, **H**, **Y**, **Z**. In this example, the operator $\vee$ denotes logical disjunction, and can be used in goals and conditional parts of rewrite rules to express alternative solutions.

Solving this goal requires a cooperation among solvers for linear, polynomial, and differential equations over the domain of complex numbers. To solve **G**, we call

```
In[11]:= TSolve[G, {X, H, Y, Z}, {},
Rules → Prog,
DefinedSymbol → {f : Compl → Compl},
Constructor → {g : Compl → Compl}]
```

Type checking program ...

Type checking goal ...

Enabling distributed constraint solving subsystem ...

$$\{X \rightarrow 3 + \sqrt{15}, Y \rightarrow -\sqrt{15}, Z \rightarrow 4, H \rightarrow \lambda[\{x \cdot 1\}, \frac{4x \cdot 1^{1-\sqrt{15}}}{1 - \sqrt{15}} + C1_2]\}$$

$$\{X \rightarrow 3 - \sqrt{15}, Y \rightarrow \sqrt{15}, Z \rightarrow 4, H \rightarrow \lambda[\{x \cdot 1\}, \frac{4x \cdot 1^{1+\sqrt{15}}}{1 + \sqrt{15}} + C1_4]\}$$

$$\{X \rightarrow -\sqrt{9 - \sqrt{15}}, Y \rightarrow -\sqrt{15}, Z \rightarrow 4, H \rightarrow \lambda[\{x \cdot 1\}, \frac{4x \cdot 1^{1-\sqrt{15}}}{1 - \sqrt{15}} + C1_6]\}$$

$$\{X \rightarrow -\sqrt{9 + \sqrt{15}}, Y \rightarrow \sqrt{15}, Z \rightarrow 4, H \rightarrow \lambda[\{x \cdot 1\}, \frac{4x \cdot 1^{1+\sqrt{15}}}{1 + \sqrt{15}} + C1_8]\}$$

$$\{X \rightarrow \sqrt{9 - \sqrt{15}}, Y \rightarrow -\sqrt{15}, Z \rightarrow 4, H \rightarrow \lambda[\{x \cdot 1\}, \frac{4x \cdot 1^{1-\sqrt{15}}}{1 - \sqrt{15}} + C1_{10}]\}$$

$$\{X \rightarrow \sqrt{9 + \sqrt{15}}, Y \rightarrow \sqrt{15}, Z \rightarrow 4, H \rightarrow \lambda[\{x \cdot 1\}, \frac{4x \cdot 1^{1+\sqrt{15}}}{1 + \sqrt{15}} + C1_{12}]\}$$

$$\begin{aligned} Out[11] = & \left\{ \left\{ X \rightarrow 3 + \sqrt{15}, Y \rightarrow -\sqrt{15}, Z \rightarrow 4, H \rightarrow \lambda[\{x \cdot 1\}, \frac{4x \cdot 1^{1-\sqrt{15}}}{1 - \sqrt{15}} + C1_2] \right\}, \right. \\ & \left\{ X \rightarrow 3 - \sqrt{15}, Y \rightarrow \sqrt{15}, Z \rightarrow 4, H \rightarrow \lambda[\{x \cdot 1\}, \frac{4x \cdot 1^{1+\sqrt{15}}}{1 + \sqrt{15}} + C1_4] \right\}, \\ & \left\{ X \rightarrow -\sqrt{9 - \sqrt{15}}, Y \rightarrow -\sqrt{15}, Z \rightarrow 4, H \rightarrow \lambda[\{x \cdot 1\}, \frac{4x \cdot 1^{1-\sqrt{15}}}{1 - \sqrt{15}} + C1_6] \right\}, \\ & \left\{ X \rightarrow -\sqrt{9 + \sqrt{15}}, Y \rightarrow \sqrt{15}, Z \rightarrow 4, H \rightarrow \lambda[\{x \cdot 1\}, \frac{4x \cdot 1^{1+\sqrt{15}}}{1 + \sqrt{15}} + C1_8] \right\}, \\ & \left\{ X \rightarrow \sqrt{9 - \sqrt{15}}, Y \rightarrow -\sqrt{15}, Z \rightarrow 4, H \rightarrow \lambda[\{x \cdot 1\}, \frac{4x \cdot 1^{1-\sqrt{15}}}{1 - \sqrt{15}} + C1_{10}] \right\}, \\ & \left. \left\{ X \rightarrow \sqrt{9 + \sqrt{15}}, Y \rightarrow \sqrt{15}, Z \rightarrow 4, H \rightarrow \lambda[\{x \cdot 1\}, \frac{4x \cdot 1^{1+\sqrt{15}}}{1 + \sqrt{15}} + C1_{12}] \right\} \right\} \end{aligned}$$

In this case, the solutions computed by CFLP are parametric, and the system has generated the auxiliary variables $C1_7$, $C1_8$, $C1_9$, $C1_{10}$, $C1_{11}$, $C1_{12}$ to express them. It is not hard to see that these are all the solutions of $\mathbf{G}$.

■ The Structure of the System

CFLP is a distributed software system for solving systems of equational goals in theories that can be represented as sets of conditional rewrite rules over a term algebra whose signature is extended with external operators. The external operators are used for expressing constraints over various domains.

The system consists of three components:

- an interpreter based on a higher-order lazy narrowing calculus C
- a cooperative constraint solver consisting of
 - a scheduler which implements a strategy S for solver cooperation
 - various specialized constraint solvers

The general architecture of the system is depicted below.

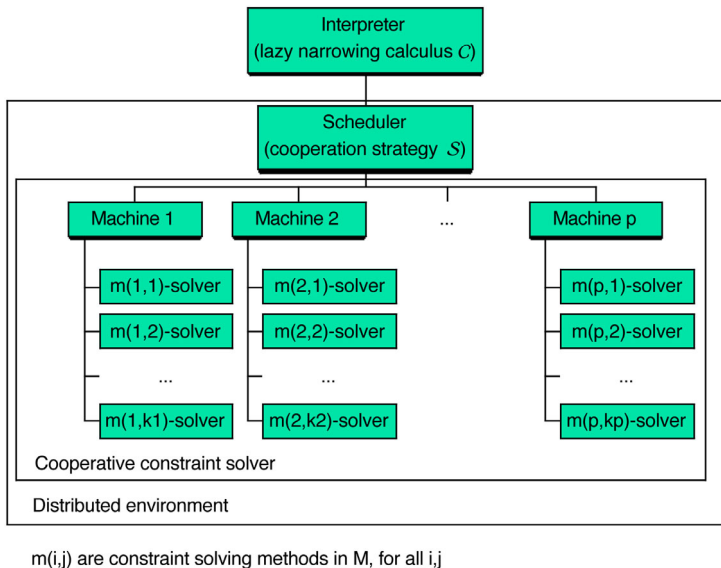


Figure 1. The architecture of CFLP.

□ The Interpreter

The CFLP interpreter is designed to solve systems of equations between simply-typed λ -terms in theories axiomatized by finite sets of conditional rewrite rules of a certain kind, called *pattern rewrite rules*.

Roughly speaking, simply-typed λ -terms are analogous to the **Function** construct of *Mathematica*, modulo the following differences.

1. The keyword **Function** is replaced with the keyword λ .

2. Each such construct (which we will henceforth call a λ -term) should be well-typed, that is, a type can be inferred from the types of its base components (variables, constants, and function symbols).

The current version of CFLP recognizes the following base types:

Type	Literals
Int	Integer literals of <i>Mathematica</i>
Float	Real literals of <i>Mathematica</i>
Complex	Complex literals of <i>Mathematica</i>
String	String literals of <i>Mathematica</i>
Bool	True, False

and the following constructed types:

$\tau_1 \times \dots \times \tau_n \rightarrow \tau$	type of functions with n arguments of types $\tau_1, \dots, \tau_n$, and result of type τ
TyList [τ]	type of lists with elements of type τ

For lists, we have adopted both the *Mathematica* notation $\{a_1, \dots, a_n\}$ to represent a list with elements $a_1, \dots, a_n$, and the Prolog-like notation **[H|T]** to represent a list with head H and tail T .

The CFLP interpreter is based on a higher-order lazy narrowing calculus **C** for pattern rewrite systems [3] extended to handle also conditional pattern rewrite systems [4]. The underlying calculus essentially consists of the unification rules of higher-order patterns [5] and the lazy narrowing rules. Since the search space of such a calculus is extremely large, we have implemented a couple of refinements with smaller search space for solutions, which are sound and complete for classes of functional logic programs and of practical interest.

We have extended these calculi in two main directions:

1. the possibility to specify *constraints*, that is, equations that cannot be solved by narrowing, but for which specialized solvers are available; and
2. the possibility to specify explicit OR-parallelism.

The interpreter successively decomposes the goal toward an answer substitution by applying the inference steps of the underlying lazy narrowing calculus. The only equations that cannot be solved in this way are the constraints. The constraints generated upon derivations are collected and sent to be solved by specialized constraint solvers via the component called the *scheduler* (see Figure 1).

Note that the nondeterministic selection of a rewrite rule for a defined symbol and the explicit OR-formulas cause the initial goal to be reduced to disjoint sets

of constraints that can be solved in parallel. Thus, in the last illustrative example, the reduction of the initial goal

$$f[X] \approx g[Y], g[Y^2] \approx g[Z^2 - 1], \\ \lambda[\{x : \text{Compl}\}, H'[x]] \approx \lambda[\{x : \text{Compl}\}, Zx^Y], H'[1] = 4$$

involves the decomposition of the equation $f[X] \approx g[Y]$ into simpler equations. The transformation step performed by our lazy narrowing calculus is

$$f[X] \approx g[Y] \Rightarrow X \approx X0, g[Y0] \approx g[Y], (X0 + Y0 = 3 \vee X0^2 - Y0 = 9) \quad (2)$$

where $X0, Y0$ are new variables. In this step we have used the fresh variant

$$f[X0] \rightarrow g[Y0] \Leftarrow (X0 + Y0 = 3 \vee X0^2 - Y0 = 9)$$

of the rewrite rule which defines f .

During step (2), an OR-subgoal is produced and as a result the initial goal is finally decomposed into two disjoint sets of constraints. These sets of constraints are sent to be solved to the scheduler.

□ The Scheduler

The scheduler coordinates the process of solving the systems of constraints received from the interpreter. In order to solve these sets of constraints, the constraint scheduler maintains a dynamic data structure called a *constraint tree*. The nodes of the constraint tree are pairs of the form $\langle \theta, \varsigma \rangle$ where θ is a substitution and ς is a set of constraints.

Whenever a set of constraints ς is received from the interpreter, the scheduler adds a new son $\langle \epsilon, \varsigma \rangle$ to the root of the constraint tree. Here ϵ stands for the empty substitution. The scheduler expands this tree by applying constraint-solving methods in parallel to its leaf nodes.

A leaf node $\langle \theta, \varsigma \rangle$ is expanded with respect to a method m as follows.

1. ς is decomposed into a set ς_1 of constraints to which method m can be applied, and a set ς_2 of other constraints.
2. ς_1 is sent to be solved to a constraint solver which implements method m . We call such a solver an *m-solver*.
3. If the *m-solver* detects ς_1 inconsistent, then the node $\langle \theta, \varsigma \rangle$ is marked as inconsistent. Otherwise the *m-solver* returns $\langle \epsilon, \varsigma \rangle$ if it cannot reduce ς_1 , or it computes a finite sequence of pairs $\langle \theta_1, \varsigma'_1 \rangle, \dots, \langle \theta_p, \varsigma'_p \rangle$ with the property that θ is a solution of ς if and only if there exists a solution σ_i of some ς'_i ($1 \leq i \leq p$), such that $\theta = \theta_i \sigma_i$.
4. If the sequence $\langle \theta_1, \varsigma'_1 \rangle, \dots, \langle \theta_p, \varsigma'_p \rangle$ is computed by the *m-solver*, then the nodes $\langle \theta \theta_i, \varsigma_2 \theta_i \cup \varsigma'_i \rangle$ ($1 \leq i \leq p$) are added to the constraint tree as sons of the node $\langle \theta, \varsigma \rangle$.

A node $\langle \theta, \varsigma \rangle$ is *final* if it cannot be reduced by any *m-solver* which is available.

The implementation of the scheduling algorithm is inspired by the work of Hong [6]. The scheduler can be regarded as a component parameterized with a list $\mathbf{M} = \{m_1, \dots, m_k\}$ of constraint-solving methods. The scheduler implements a cooperation strategy $\mathbf{S}$, which repeatedly applies the sequence $m_1, \dots, m_k$ of methods to the leaves of the constraint tree until they become either final or inconsistent. As soon as a final node is generated, it is made accessible to the interpreter.

□ **The Constraint Solvers**

The constraint solvers are implementations of the constraint-solving methods specified to the scheduler through the list $\mathbf{M}$. The current implementation of CFLP provides constraint solvers, which implement the following methods for solving constraints over the domains of real and complex numbers.

Linear	a method for solving systems of linear equations and systems of equations with invertible functions
Polynomial	a method for solving systems of polynomial equations
Derivative	a method for solving derivative equations
PartialDerivative	a method for solving partial derivative equations

These methods are tried in the order presented.

All solvers are implemented by separate *Mathematica* processes executing in parallel and communicating with the constraint scheduler via *MathLink* connections. There are two types of constraint solvers in CFLP.

1. Local constraint solvers that run as subsidiary *Mathematica* processes of the CFLP constraint scheduler.
2. Shared constraint solvers are started from outside a CFLP session and can be connected later to more CFLP schedulers, which may run on a different machine.

The user can adjust the distributed constraint-solving component of the system by specifying the number of local constraint solvers that are started at system initialization, and the remote machines on which to look for shared constraint solvers.

The communication mechanism between scheduler and constraint solvers is implemented completely in *MathLink* [7]. Therefore, CFLP is a platform-independent software system and can be used in heterogeneous networks.

■ **Conclusions and Future Work**

CFLP is a software system consisting of a functional logic interpreter and a distributed constraint-solving system, which provides support for solving systems

of equations over various constraint domains. The functional logic component allows the user to define his own abstractions in an easy and comfortable way. In the current implementation we have integrated solvers for linear equations and equations with invertible functions, polynomial equations, differential equations, and partial differential equations. The constraint solvers are all implemented on top of the constraint-solving capabilities of *Mathematica*.

We intend to further develop the system by integrating more constraint-solving capabilities. The constraint solvers may act either on disjoint sets of constraints or on overlapping subsystems. Currently, the constraint solvers are not allowed to act simultaneously on a leaf node of the constraint tree. An optimization would be to act simultaneously with more solvers on the same node in situations when the subsystems of equations are non-overlapping.

Because of the higher-order extensions of the functional logic programming framework (function variables and λ -abstractions), the search space of lazy narrowing is very large. We have identified and implemented various refinements of a lazy narrowing calculus that perform a more deterministic search for solutions for classes of functional logic programs of practical interest. We intend to continue our research and identify better refinements.

The system is intended to be used by researchers in functional and logic programming languages, and by researchers in constraint solving who are willing to make use of its expressive and computational power.

■ References

- [1] J. Jaffar and J.-L. Lassez, *Constraint Logic Programming*, Technical Report 86-74, Department of Computer Science, Monash University, Clayton, 1986.
- [2] M. Marin, T. Ida, and W. Schreiner, "A Distributed System for Solving Equational Constraints based on Lazy Narrowing Calculi," in *JSSST Workshop on Programming and Programming Languages (PPL '99)*, Atagawa, Japan, March 17-19, 1999 pp. 67-78.
- [3] M. Marin, T. Ida, and T. Suzuki, "On Reducing the Search Space of Higher-Order Lazy Narrowing," in *Proceedings of the 4th Fuji International Symposium on Functional and Logic Programming, FLOPS '99, LNCS 1722*, (A. Middeldorp, T. Sato, eds.), Berlin and Heidelberg: Springer-Verlag, 1999 pp. 319-334.
- [4] C. Prehofer, *Solving Higher-order Equations. From Logic to Programming*, Boston: Birkhäuser, 1998.
- [5] T. Nipkow, "Functional Unification of Higher-order Patterns," in *Proceedings of the 8th IEEE Symposium on Logic in Computer Science*, Los Alamito, CA: IEEE Computer Society Press, 1993 pp. 64-74.
- [6] H. Hong, "RISC-CLP(CF): Constraint Logic Programming over Complex Functions," in *Proceedings of the 5th International Conference on Logic Programming and Automated Reasoning, LPAR '94*, (F. Pfennig ed.), Berlin Heidelberg: Springer-Verlag, 1994 pp. 99-113.
- [7] S. Wolfram, *The Mathematica Book*, 3rd ed., Champaign: Wolfram Media and Cambridge: Cambridge University Press, 1996.

■ Additional Material

Available at www.mathematica-journal.com.

cflp.tar.gz

See also www.score.is.tsukuba.ac.jp/software/CFLP

About the Authors

Doctor Mircea Marin is a JSPS postdoctoral fellow at the Institute of Information Sciences and Electronics, University of Tsukuba, Japan. His recent work includes the design and implementation of distributed systems for solving systems of constraints in a higher-order setting by using CORBA middleware, mathematical libraries, and computer algebra systems, such as *Mathematica*. For details on his recent work, see www.score.is.tsukuba.ac.jp/~mmarin.

Doctor Tetsuo Ida is a professor at the Institute of Information Sciences and Electronics, University of Tsukuba, Japan, where he leads a research group on symbolic computation. For many years he has worked on various aspects of symbolic computation, such as rewrite theories, parallel hashing algorithms, and Lisp Machine for symbolic computation. He is an editor of the *Journal of Symbolic Computation*. For details on his recent work, see www.score.is.tsukuba.ac.jp/~ida.

Doctor Wolfgang Schreiner is an assistant professor for the Research Institute for Symbolic Computation (RISC-Linz) at the Johannes Kepler University in Linz, Austria. He has worked for several years on parallel and distributed systems for symbolic and algebraic computation using various declarative languages, mathematical libraries, and computer algebra systems. For details on his recent work, see www.risc.uni-linz.ac.at/people/schreine.

Behavior and Learning under Law: Automata Containing Evolutionary Algorithms

Chapter 1. Tool Construction

Seth J. Chandler

*Associate Professor of Law
University of Houston Law Center
Health Law Suite 104H
100 Law Center
Houston, Texas, 77204, USA
SCChandler@uh.edu*

Christian J. Jacob

*Assistant Professor
Department of Computer Science
University of Calgary
2500 University Drive NW
Calgary T2N 1N4, Canada
jacob@cpsc.ucalgary.ca*

This article studies the evolution of cellular automata in which the cells each contain data and genetic programs. The immediate application of this convergence is to study how different legal rules affect the evolution of learning and behavior in an economy. The study uses the typical “Coase Theorem” scenario, in which neighbors engage in activities that may detrimentally affect each other.

Originally presented at the Third International *Mathematica* Symposium (Hagenberg, Austria, August 23–25, 1999).

■ Introduction

To date, the study of cellular automata and genetic programming have proceeded largely on parallel paths. This article studies the evolution of cellular automata in which the cells each contain data and genetic programs. The immediate application of this convergence is to study how different legal rules affect the evolution of learning and behavior in an economy. Drawing on the disciplines of computer science, biology, law, and economics, the article provides a new “artificial intelligence” tool for use in economic modeling. The typical “Coase Theorem” [1] scenario, in which neighbors engage in activities that may detrimentally affect each other, is to explore some of the capabilities of the new tool [2].

The authors use the *Mathematica* programming language to implement their conception. They create a ring of sites, each of which contain genetic programs [3] for determining the behavior and learning strategy of that site. These genetic programs are created using a flexible template mechanism that facilitates specification of a complex grammar for permissible programs, as was demonstrated for the evolution of Lindenmayer systems and plant growth programs [4]. The rich set of primitives of these programs include sensing operations, such as determination of the behaviors of neighboring sites and mating operations that permit programs of neighboring sites to be adopted in part or in their entirety. These primitives thus permit the sites to learn from each other and, in a deeply recursive manner, to learn how the other sites themselves learn. This is a novel approach of combining evolutionary algorithms and, in particular, genetic programming techniques [5, 6, 7] with interacting agents modeled by cellular automata.

The ring of sites (the economy) evolves through a multistage iteration. First, the programs of the sites are evaluated in a cascading fashion to determine the future course of learning by the site, the immediate program for determining behavior, and the actual behavior resulting from the interaction of the behavior program with local and global features of the economy. The behaviors are in turn evaluated to determine certain global parameters (such as the prices of various commodities within the economy) and the consequences of the local interactions created as a result of the behaviors and the legal rules applicable thereto. This process permits a computation of the fitness of each of the sites within the economy. The final state of each evolutionary iteration permits the programs at each site to mutate based in part on the fitness of the site. The coevolution of these sites is modeled through the repetition of these iterations. The emphasis in this paper is on the development and validation of the tools necessary for this form of modeling and not on particular results.

■ Background

To understand this article, it is necessary to have a brief background in both cellular automata and genetic programs.

□ Cellular Automata

Homogeneous Cellular Automata

Conventional homogeneous cellular automata begin with a group of sites, each one of which has an identical structure of neighbors. Each site has some value, which is usually, though not necessarily, an integer often expressed in base 2. Thus a site might have the value 10001 represented in base 2 (17 in base 10). That value evolves according to some exogenously-imposed global rule that uses only the value of each site and some subset of that site's neighbors as its data. Traditionally, the subset of sites is the site itself and those within some distance (or radius) r of the site. The rule is the same for all the sites, hence the term "homogeneous cellular automata". [Note1]

In a cellular automaton, the value of all sites evolves synchronously. A ring in which each site has one neighbor on its left and one neighbor on its right, in which the value of each site can be 0 or 1, and in which the new value of the site is equal to the mod 2 sum of the site and these neighbors is a conventional cellular automaton. Figure 1 below illustrates such a cellular automaton with twelve sites on the ring through ten iterations. Position 1 appears at "1 o'clock" and position 12 appears at "12 o'clock". Iterations 1–5 appear left to right in the top row. Iterations 6–10 appear left to right in the second row. [Note2]

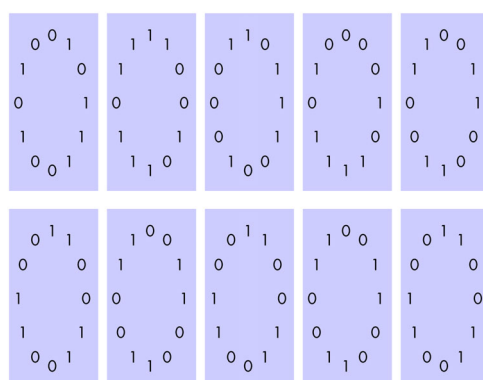


Figure 1. The evolution of a homogeneous cellular automaton on a ring.

The evolution can be viewed in an alternative way by (1) coloring sites whose value is 1 as white and coloring sites whose value is 0 as black, (2) "snipping" each of the rings at some arbitrary point (such as between 12 o'clock and 1 o'clock), (3) displaying each of the snipped rings as a colored horizontal line whose beginning and end we know to be connected, and (4) putting lines representing later stages of evolution below those representing earlier stages of evolution. The evolution shown in Figure 1 can also be displayed as in Figure 2.

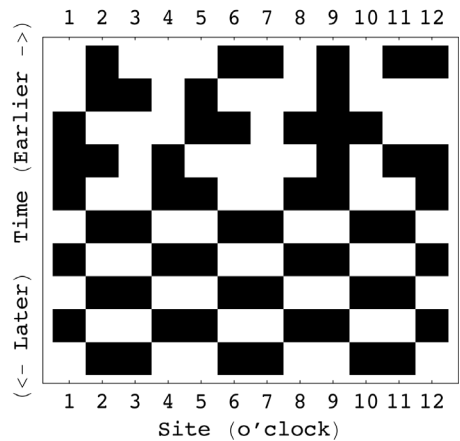


Figure 2. The evolution of an homogeneous cellular automaton with periodic boundary conditions.

Numbering Rules

The rule used to evolve an automaton can be given an integer representation in a way pioneered by Wolfram [8]. The rule discussed above, in which the new value is the mod 2 sum of the site and its left and right neighbors, creates the following map between the neighborhood of a site and the new value at that site. [Note3]

{1, 1, 1}	→ mod[3, 2]	→ 1
{1, 1, 0}	→ mod[2, 2]	→ 0
{1, 0, 1}	→ mod[2, 2]	→ 0
{1, 0, 0}	→ mod[1, 2]	→ 1
{0, 1, 1}	→ mod[2, 2]	→ 0
{0, 1, 0}	→ mod[1, 2]	→ 1
{0, 0, 1}	→ mod[1, 2]	→ 1
{0, 0, 0}	→ mod[0, 2]	→ 0

If we now go down the right-most element of each line of the above table, we see 10010110. That sequence is a representation of the rule employed to evolve the automaton. We can think of this sequence as the binary number 10010110₂, which has a decimal equivalent of 150. The rule described above is thus often given the shorthand of “Rule 150”. How many distinct rules can there be? Since it takes eight binary digits to represent a rule, there are 256 distinct possible rules. Generally, if the rule requires examination of x digits represented in base b and outputs a digit in base b , it takes b^x digits in base b to represent the rule and there are thus b^{b^x} possible rules. When b or x is large, this number can become amazingly large.

Inhomogeneous Cellular Automata

Inhomogeneous cellular automata also exist, in which each site evolves according to its own rule [9]. The site at the “first” position in an automaton might evolve according to Rule 150, whereas the site at the second position might evolve

according to Rule 129, and the site in the third position might evolve according to Rule 226, and so forth. Figure 2 illustrates the evolution of such an inhomogeneous cellular automaton in which the initial state of the automaton is the same as the initial state of the automaton shown in Figure 1, but in which the sites respectively employ {150,129,226,14,154,158,141} as the rules for evolution.

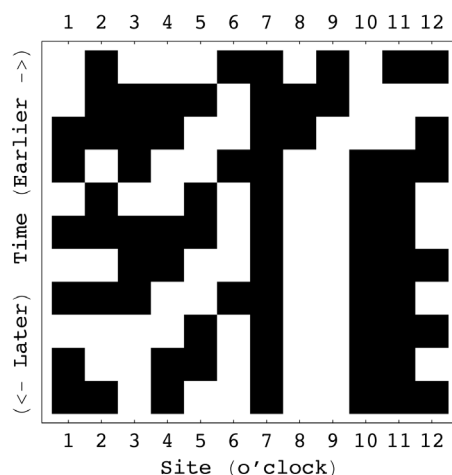


Figure 3. The evolution of an inhomogeneous cellular automaton with periodic boundary conditions.

One may think of this process as evolving behavior based on the individual site's prespecified mechanism for updating itself [10]. The sites thus react in their own ways to the behavior around them, but do not evolve in the ways in which they react to that behavior.

Converting Inhomogeneous Cellular Automata into Homogeneous Cellular Automata

An inhomogeneous cellular automaton, however, can be recharacterized as a homogeneous cellular automaton. In the example above, the value at each site can be considered as a nine digit binary number, the first digit of which represents what we formerly called the value of the site and the last eight digits of which represents what we called the rule for evolving that site. Where a site has value 1 and uses Rule 150 (10010110_2) to update itself, the equivalent homogeneous cellular automaton has a value of 406 or 110010110_2 . Generally, the value of a site is recharacterized as a now indistinguishable combination of what we formerly called the value of the site and the rule for updating that site. What one might call the data and the program have been merged into pure data.

In this homogeneous recharacterization of an inhomogeneous cellular automaton, the first digit of a site can vary, while the last eight digits (the rule part) cannot. To evolve, each site consults the first digit of the site to the left, all of its own nine digits, and the first digit of the site to the right. There is then a global rule—a sort of metaprogram—which itself can be characterized as a 2048

(2^{1+9+1}) binary digit number that sets forth how the first digit of each new site is to be updated. In our example, the value of that rule in decimal form is about $3.231700607131101 \times 10^{616}$.

Generally, if each site in an inhomogeneous cellular automaton can take on k values and updates itself by looking at itself and sites within some radius r , it takes $k^{2^{r+1}}$ base k digits to specify an update rule and thus takes $k^{k^{2^{r+1}}+2^{r+1}}$ digits to specify the global update rule of the equivalent homogeneous cellular automaton. Obviously, for each homogeneous cellular automaton, there is an equivalent inhomogeneous automaton, with all the cells using the same update rule.

Second-Level Inhomogeneous Cellular Automata and Beyond

One can take the notion of inhomogeneity to the next level as well. Each site cannot only have a value and a rule for updating itself, but it can also have a rule for updating the rule by which it then updates itself. The site evolves its behavior based on the behavior of the neighboring sites, but does so in a way that itself evolves. The site thus exhibits what might be called learning. At any given time in our sample automaton, each site would have a one-digit value and an eight-digit update rule code. It would also have a fixed and very lengthy code for updating the update code based on the values at the neighboring sites and their update codes. Indeed, it appears that a complete specification of the code for updating the update code would take 2^{30} (about one billion) binary digits. [Note4]

The resulting inhomogeneous cellular automaton can, like all cellular automata, be converted into a homogeneous cellular automaton, but one with an almost inconceivably long global update rule.

Of course, at a theoretical level, there is no need to stop at second-level inhomogeneity. We might want the site to learn how to learn by updating the code by which it updates its update code. Each site would first examine the $2^{30} + 9$ digits of itself and its three neighbors, and use an update rule with $2^{3221225529}$ digits (this is 2 to about the three trillionth power) to update the 2^{30} digit update code used for updating the update rules used for revising the values. With the rules for updating the update codes in place, the automaton would then update the update codes. And with the update codes in place, the automaton would then update the values.

In mathematical theory, this process of recursion can continue indefinitely, with the lengths of the various update codes surpassing imagination. As a practical matter, however, no imaginable computer can handle this sort of data—we are well past the number of known particles in the universe—in any imaginable time. Hence, modeling learning and learning how to learn needs to employ a different mechanism than simply a raw specification of update codes. [Note5]

□ Genetic Programs

Genetic programs are a form of computer programs that evolve. The programs themselves have a structure in which the permissible arguments to each function are determined by a specified “grammar”. Certain of the programs may “mate” with each other by permitting certain portions of one program to replace por-

tions of another program. The points at which the mating programs “cross over” are determined by a “mating template”. The programs may also mutate spontaneously.

Construction of Random Expressions

A set of notebooks developed by Jacob permits *Mathematica* to generate random expressions from a particular grammatical structure. The user creates this grammatical structure by writing a list of correct expression patterns (templates) that are themselves *Mathematica* expressions. The process may be easier to demonstrate than describe. Code Fragment 1 shows an example.

```
basicgrammar = {sentence[_noun, _verbphrase, _noun],
  sentence[_noun, _verbphrase],
  sentence[_sentence, _conjunction, _sentence],
  noun[_article, _thing], verbphrase[_verb, _adverb],
  verbphrase[_verb], article[the], article[a], verb[hit],      (1)
  verb[examine], verb[eat], verb[smell], thing[boy],
  thing[gorilla], thing[plant], thing[cat], thing[girl],
  adverb[quickly], adverb[softly], adverb[really],
  adverb[violently], conjunction[while], conjunction[but]};
```

A sentence consists of either a noun followed by a verb phrase followed by a noun, a noun followed by a verb phrase, or a sentence followed by a sentence followed by another sentence. A noun might in turn consist of an article followed by a thing. A verb phrase might consist of either a verb followed by an adverb or just a verb. A verb might be hit, examine, eat, or smell. A thing might be boy, gorilla, plant, cat, or girl. An adverb might be quickly, softly, really, or violently.

If we evaluate a notebook created by Jacob called GP-ExprGeneration.nb, the user may then input this grammar as an argument to one of several functions such as **createExpression** that then generate grammatically correct expressions at the desired level of complexity. Code Fragment 2 illustrates the process, creating a list of three sentences.

```
mysentences =
  Table[createExpression[
    sentence[_noun, _verbphrase, _noun, while, _sentence],      (2)
    ExprPatterns → basicgrammar, MaxDepth → 18], {3}]

{sentence[noun[article[a], thing[gorilla]],
  verbphrase[verb[smell], noun[article[the], thing[gorilla]],
  while, sentence[noun[article[the], thing[boy]],
  verbphrase[verb[examine], noun[article[a], thing[girl]]]],
  sentence[noun[article[the], thing[cat]],
  verbphrase[verb[examine], adverb[softly]],
  noun[article[the], thing[cat]], while,
  sentence[noun[article[a], thing[boy]],
  verbphrase[verb[smell], noun[article[a], thing[plant]]]],
  sentence[noun[article[a], thing[plant]], verbphrase[verb[hit]],
  noun[article[the], thing[cat]], while,
  sentence[noun[article[a], thing[gorilla]],
  verbphrase[verb[examine], adverb[quickly]]]}}
```

We can reformat this output to make it a bit more readable.

```
{a gorilla smell the gorilla while the boy examine a girl ,
  the cat examine softly the cat while a boy smell a plant ,
  a plant hit the cat while a gorilla examine quickly }
```

While these sentences may not make a great deal of sense, they represent permissible grammatical constructions.

The Jacob method for generating random symbolic expressions can also be used to generate structures with more apparent mathematical meaning. Code Fragment 3 is a minimal grammar of expressions that look a lot like *Mathematica* expressions.

```
mathgrammar = {power[_ , _plus | _times | 2 | 3], plus[_ , _],
  times[-1, _], x, 2, 3};
```

 (3)

In Code Fragment 4 we show how to create five random expressions using this grammar and the `createExpression` function.

```
createExpressions[5, _power, ExprPatterns → mathgrammar,
  MaxDepth → 6, Trials → 30, SequenceRange → {1, 2}];
```

 (4)

Here are some sample results.

```
mathexpressions = {power[3, 2], power[x, 2],
  power[power[x, times[-1, 2]], 2], power[times[-1, x], 3],
  power[times[-1, x], plus[times[-1, 2], 2, x]]};
```

 (5)

These expressions can be visualized as trees.

```
power[3, 2]
power[x, 2]
power[ | , 2]
      power[x, | ]
              times[-1, 2]
power[ | , 3]
      times[-1, x]
power[ | , | ]
      times[-1, x] plus[ | , 2, x]
                      times[-1, 2]
```

Mating of Expressions

In a process analogous to mating in the “real world”, expressions created in this grammar can be mated to each other by permitting any permissible component of the grammar in one expression to be replaced by any component of the mate that matches another (possibly different) component of the grammar. The computer code is treated analogously to biological material capable of evolution. Essentially, we snip off fragments from one expression that match a pattern and graft on fragments from a second expression that match a second (possibly different) pattern. If more than one fragment on each tree matches, we select randomly from the potential snip and graft candidates. The *Mathematica* code for this procedure of recombination is shown below in Code Fragment 6.

```

randommateexpressions[me_, mate_, mypattern_, matepattern_] :=
  With[
    {mepos = With[{q = Position[me, mypattern]},
      Part[q, Random[Integer, {1, Length[q]}]]],
    matepos = With[{q = Position[mate, matepattern]},
      Part[q, Random[Integer, {1, Length[q]}]]],
    If[matepos != {}, ReplacePart[me, mate, {mepos}, {matepos}], me]]

```

(6)

For example, we can mate the third expression in **mathexpressions** to the fourth expression in **mathexpressions** allowing any subexpression with a head of **Integer** in the first expression to be replaced by any subexpression with a head of **Times** in the second expression. We do this six times to illustrate the different possible results. [Note6]

```

Table[randommateexpressions[mathexpressions[[4]],
  mathexpressions[[5]], _Integer, _times], {6}]

```

(7)

$$\{\text{power}[\text{times}[\text{times}[-1, x], x], 3], \text{power}[\text{times}[-1, x], \text{times}[-1, 2]],$$

$$\text{power}[\text{times}[\text{times}[-1, x], x], 3], \text{power}[\text{times}[-1, x], \text{times}[-1, 2]],$$

$$\text{power}[\text{times}[-1, x], \text{times}[-1, 2]], \text{power}[\text{times}[-1, x], \text{times}[-1, x]]\}$$

We can then evaluate these expressions using some substitution rules in conjunction with the **ReplaceRepeated** construct.

```

ReplaceRepeated[% , {plus → Plus, power → Power, times → Times}]

```

(8)

$$\{-x^6, \frac{1}{x^2}, -x^6, \frac{1}{x^2}, \frac{1}{x^2}, (-x)^{-x}\}$$

A more general method of defining an extended set of genetic operators on symbolic expressions, such as recombination, inversion, deletion, duplication, and mutation, is described in detail in [7]. Further code examples are available at www.cpsc.ucalgary.ca/~jacob/IEC.

□ Conceptually Combining Automata with Genetic Programs

While inhomogeneous cellular automata can in theory be used to model agents that learn and that learn how to learn, as noted above, there is a practical impediment: the amount of data needed to characterize the rules for learning exceeds the size of any conceivable computer, even for relatively simple automata. One way around this is to permit a site to hold both data and a computer program for updating itself. The computer program references itself and the data surrounding it to figure out how to update the site. If the set of operations in the computer program is broad enough, this method should be capable of spanning a broad set of the possible rules for learning and learning how to learn. One way to create a rich set of operations is through genetic programs.

■ The Model

□ Motivation for the Model

This combination of genetic programming and cellular automata emerges out of the desire to study a classic problem in the economic analysis of law: conflicting

use of adjoining land use [1]. The problem arises in which the behavior of one landowner affects the profitability of neighboring landowners. Thus the production of vegetable farmers may be adversely affected by the number of cows maintained on an unfenced landscape by neighboring ranchers.

Traditional economic analysis of this problem has assumed that landowners have astonishing powers of computation, anticipation, and observation, and that they assimilate all local and global properties of the economic and legal system. It further assumes that all adjustments to equilibrium take place instantaneously. These models fail to address more realistic settings in which agents interact locally and learn on the basis of limited knowledge regarding the local and global environment.

One departure from the traditional method of analysis was made earlier by Chandler [2], in which he assumed that actors could select learning mechanisms and mechanisms for learning how to learn from a set of prespecified choices that the author thought relatively logical. That work, like this one, explored how changing the rules of law altered both behaviors and the methods of learning employed by agents in the economy. While the results of that work were intriguing—the emergence of order roughly paralleling cooperative behavior—the constraint on choices available to the agents was somewhat artificial.

In this paper, that constraint is relaxed. While the grammar from which possible choices are to be made is constrained—as it is in any genetic programming approach—the possible constructions from that grammar are essentially unlimited. The agents are thus given far greater freedom to explore. The idea of this paper is to see what sorts of learning and behaviors in fact emerge, and in what period of time, given this freedom and the background rules of law. One issue to be explored is the extent to which changes in the rule of law changes methods of learning, forms of behavior, and gross features of the economy.

□ The Economy

Site Activities

Each site in this economy either produces cows (ranching) or grows vegetables (farming). The amount of production at each site varies from 0 to 7. Farmers growing vegetables may have all or part of their crop destroyed by cows from adjacent properties. If c_L is the number of cows at the site to the left of the farmer and c_R is the number of cows at the site to the right of the farmer, then the vegetables remaining (**vpost**) after trampling is **Max**[0, **Round**[$(-1 + a c_L) (-1 + a c_R) v$]], where v is the level of production of vegetables by the farmer and a is a constant representing the destructive force of surrounding cows.

Cost of Production

It costs money to produce either cows or vegetables. The formula (**tc**) for total cost of production of either vegetables or cows is $1 + 9x - \frac{43}{16}x^2 + \frac{11}{32}x^3$, where x is the quantity produced. This formula results in long-run equilibrium being attained when the price is equal to four. [Note7]

Demand

The demand function in the economy is determined by the number of sites (σ) in the economy. If p is the price of a commodity (either cows or vegetables), then the quantity demanded is $\frac{\sigma}{2}(\delta - p)$. [Note8]

Prices

Consistent with traditional economic learning, the price of cows and vegetables in the economy is determined by the intersection of endogenously determined supply at a particular time with the current demand. For example, if there are 20 sites in the economy, and δ is set equal to 8, the demand for cows is thus $20(8 - p)$. If the quantity of cows supplied is 63, then the equilibrium price is $\frac{97}{20}$ because $20(8 - \frac{97}{20}) = 63$.

□ The Data Structures Employed to Model the Economy

We model an economy in the *Mathematica* language as a data structure called an **automatonPattern** with two parts: a **List** containing a number (σ) of sites, and a sequence of **Rule** objects containing certain global data about the economy, such as commodity prices and demand levels. Code Fragment 10 shows how this is done in *Mathematica*. The **List** is generally treated as a ring with programs treating the right end of the **List** as being adjacent to the left end. The authors accomplish this treatment through use of part indices computed as mod the length of the ring offset by one to compensate for *Mathematica*'s use of part 1 rather than part 0 as the first element of an object.

A site is in turn modeled as a **List**, which contains data on the behavior and profitability of the site (represented as an object with a head of **List**) as well as a number of programs. These programs include programs for how to behave, how to modify how to behave, how to modify how to modify how to behave, how to modify how to modify how to modify how to behave, and how to modify how to modify how to modify how to modify how to behave. The programs have a head of either **recombination** or **behavior**. Fragment 9 shows how this is done in *Mathematica*.

```
sitePattern = {_recombination, _recombination, _recombination,
               _recombination, behaviorProgram_behavior,
               {farmer | rancher, intendedIntensity_Integer?
                (Function[x, 0 ≤ x ≤ 7])},
               actualIntensity_Integer?(Function[x, 0 ≤ x ≤ 7])},
               profit_, cumulativeprofit_, other___};
```

(9)

```
automatonPattern =
  automaton[r : List[___?(Function[MatchQ[#, sitePattern]])],
  globals___Rule];
```

(10)

A site might thus look like the following.

```
{recombination[none],
 recombination[direction[right], thedirection,
  threebitint[modplus[threebitint[2], threebitint[4]]],
  threebitint[modtimes[-1, threebitint[0]]]],
 recombination[none],
 recombination[direction[medianprofitable],
  thedirection, threebitint[7], threebitint[4]],
 behavior[{activity[rancher], threebitint[3]}],
 {rancher, 0, 7, 2.9147250282233204', 0}}
```

■ Auxiliary Functions

We also create a number of utility functions that extract or create data regarding the economy. One set of utilities extracts data from an economy or its various subcomponents. Another set inserts data into an economy or its various subcomponents. [Note9]

□ Getting Data from the Automaton

One set of utilities extracts data from the automaton. Here is an example of the code.

```
getPrices[a : automatonPattern] :=
  prices /. getGlobalData[a] /. Options[Head[a]] (11)
```

□ Setting Data in the Automaton

A second set of utilities inserts data into an economy. Here is an example of the code.

```
setPrices[a : automatonPattern, v_List] :=
  a /. Rule[prices, _] → Rule[prices, v] (12)
```

□ Total Intensities

Other utilities calculate the amount of each commodity produced by the sites in the economy. The following code provides an example.

```
totalActualIntensities[a : automatonPattern] :=
  Map[
    Plus @@ Cases[behaviorList[a], {#, _, actual_, ___} → actual] &,
    activityDomain[a]] (13)
```

■ Computation

In addition to the set of simple utility programs, we also need more complex programs for determining the actual level of production achieved by farmers in light of trampling activities by surrounding cows, a program that determines the equilibrium set of prices in the economy, and, finally, a complex program for

determining the profitability of each site in light of its intended production (pretrampling), its actual production, liability for damages, compensation for damages, and the computational cost of determining how to behave.

To compute actual intensities, we take a ring of behaviors and partition it into groups of three, each offset by one site. Each site thus lies at the center of a three-length neighborhood of sites. For each three-length neighborhood of sites thus created, we examine the pattern of interference. For example, if the site in the center of one of these neighborhoods has produced ranching behavior, we know that the site's actual production of cows is the same as its intended production; vegetables do not trample cows. On the other hand, if the site in the center of one of the neighborhoods has produced farming behavior and the site on the left produces ranching behavior, we know there is a chance that the farm's level of production will be reduced according to the destructive capabilities of the neighboring cows. The code for accomplishing this procedure is shown in Code Fragment 14.

```
computeActualIntensities[destructionfactor_][blist_List] :=
Block[{reviseProduction},
  reviseProduction[df_] [
    {leftneighbor_, self : {rancher, intended_, ___},
     rightneighbor_}] := intended;

  reviseProduction[df_] [
    {leftneighbor : {farmer, leftintended_, ___},
     self : {farmer, intended_, ___},
     rightneighbor : {farmer, __}}] := intended;

  reviseProduction[df_] [
    {leftneighbor : {rancher, leftintended_, ___},
     self : {farmer, intended_, ___},
     rightneighbor : {farmer, __}}] :=
    vpost[df][{leftintended, intended, 0}];

  reviseProduction[df_] [
    {leftneighbor : {farmer, leftintended_, ___},
     self : {farmer, intended_, ___},
     rightneighbor : {rancher, rightintended_, ___}}] :=
    vpost[df][{0, intended, rightintended}];

  reviseProduction[df_] [
    {leftneighbor : {rancher, leftintended_, ___},
     self : {farmer, intended_, ___},
     rightneighbor : {rancher, rightintended_, ___}}] :=
    vpost[df][{leftintended, intended, rightintended}];

  Map[reviseProduction[destructionfactor][#] &,
    Partition[blist, 3, 1, {2, -2}]]]
```

(14)

Prices:

To compute prices, we use *Mathematica*'s **FindRoot** command to determine numerically for each commodity the price that equates demand with the total actual amount of each good produced.

Profits:

A computation of profits at each site requires a determination of several values.

- **Computation Costs:** The act of computation is not necessarily free in this model. It is expensive to figure out, for example, how neighbors are behaving or are figuring out how to evolve themselves. In this model, the computation cost for each site is a user-specified function of the programs employed at each site. An example might be some scaling factor multiplied by the dot product of the **ByteCount** and **LeafCount** of all the programs. Both **ByteCount** and **LeafCount** are vehicles in *Mathematica* for assessing the complexity of an expression. Another example might be some scaling factor multiplied by the number of times the expression **direction[mostprofitable]** appears in the program. [Note10]
- **Production Costs:** It costs money to produce vegetables or cows. The user is permitted to specify an arbitrary function to apply to each site's level of intended production.
- **Revenues:** The site each revenue receives is equal to its actual production (after trampling damage is taken into account) multiplied by the price of the commodity produced.
- **Revenue Loss:** Sites whose actual production is less than the intended production suffer a loss equal to the price of the commodity produced multiplied by the difference between intended and actual production.
- **Compensation Received:** The user may specify a function (such as the **Identity** function) that maps revenue loss to actual compensation received by each site.
- **Damages Paid:** The user supplies a legal rule regarding how damages are to be computed, which may involve apportionment of damages suffered when cows from two surrounding neighbors damage an interior site's vegetables. A rule of proportional damages frequently used in this article holds that in such situations the ranchers pay the farmer in proportion to the number of cows they actually own. A rule of "no damages", also used in this article, holds that in such situations the ranchers pay nothing.

The profit of a site ends up being the revenues plus the compensation received, less cost of production, less cost of computation, less damages paid.

■ The Main Loop Presented in the Abstract

With these many preliminaries in place, we are now in a position to evolve our economy. We start with some arbitrary initial economy represented using the **automatonPattern** discussed above. Recall that this **automatonPattern** contains, in addition to some global data, a **List** of sites treated as a ring, each of which contain a list of programs and data on its behavior. At the outset, the behavior need not be consistent with the program or the automaton as a whole.

Starting with this initial economy, we repeatedly apply a function called **process** to the economy. Using *Mathematica*'s built-in **NestWhile** or **NestWhileList** construct, this application continues for either some specified number of iterations or until some terminal condition applies. Because the **process** function lies at the heart of this project, we offer some detail on its operation, which exemplifies a *Mathematica* "functional" programming style. [Note11]

One can imagine the ring of sites as being an array of expressions $e[i,j]$ where the i subscript denotes the position of the site in the ring (the landscape) and the j position denotes in a particular way the number of the expression at that site. In this model, $e[3,1]$ thus denotes the first expression of the third site. We order the expressions at a site so that the first expression is the deepest in the recursive string of "how to modify". That is, it is the one that tells the site how (if at all) to modify how to modify how to modify how to modify its behavior. The second expression is the one that tells the site how (if at all) to modify how to modify how to modify its behavior. The last expression at a site is the one in which the site expresses its behavior. The second to last expression at a site is the one in which the site sets forth how to determine how to behave.

In an economy with five sites and in which we permit four layers of recursion and one program for determining behavior, the structure of the economy is as follows. [Note12]

$$\begin{pmatrix} e[1, 1] & e[1, 2] & e[1, 3] & e[1, 4] & e[1, 5] & e[1, 6] \\ e[2, 1] & e[2, 2] & e[2, 3] & e[2, 4] & e[2, 5] & e[2, 6] \\ e[3, 1] & e[3, 2] & e[3, 3] & e[3, 4] & e[3, 5] & e[3, 6] \\ e[4, 1] & e[4, 2] & e[4, 3] & e[4, 4] & e[4, 5] & e[4, 6] \\ e[5, 1] & e[5, 2] & e[5, 3] & e[5, 4] & e[5, 5] & e[5, 6] \end{pmatrix}$$

We can then partition this array (called **abstractarray**) into a 5×5 array (called **partitionedabstractarray**) of 3×2 blocks such that each block contains two "adjoining" programs for each of three adjoining sites. More generally, if the original automaton has a ring with σ sites, each of which has p expressions in it, the dimensions of the resulting partitioned array is $\sigma \times (p-1) \times 3 \times 2$.

$$\begin{pmatrix} \begin{pmatrix} e[5, 1] & e[5, 2] \\ e[1, 1] & e[1, 2] \\ e[2, 1] & e[2, 2] \end{pmatrix} & \begin{pmatrix} e[5, 2] & e[5, 3] \\ e[1, 2] & e[1, 3] \\ e[2, 2] & e[2, 3] \end{pmatrix} & \begin{pmatrix} e[5, 3] & e[5, 4] \\ e[1, 3] & e[1, 4] \\ e[2, 3] & e[2, 4] \end{pmatrix} & \begin{pmatrix} e[5, 4] & e[5, 5] \\ e[1, 4] & e[1, 5] \\ e[2, 4] & e[2, 5] \end{pmatrix} & \begin{pmatrix} e[5, 5] & e[5, 6] \\ e[1, 5] & e[1, 6] \\ e[2, 5] & e[2, 6] \end{pmatrix} \\ \begin{pmatrix} e[1, 1] & e[1, 2] \\ e[2, 1] & e[2, 2] \\ e[3, 1] & e[3, 2] \end{pmatrix} & \begin{pmatrix} e[1, 2] & e[1, 3] \\ e[2, 2] & e[2, 3] \\ e[3, 2] & e[3, 3] \end{pmatrix} & \begin{pmatrix} e[1, 3] & e[1, 4] \\ e[2, 3] & e[2, 4] \\ e[3, 3] & e[3, 4] \end{pmatrix} & \begin{pmatrix} e[1, 4] & e[1, 5] \\ e[2, 4] & e[2, 5] \\ e[3, 4] & e[3, 5] \end{pmatrix} & \begin{pmatrix} e[1, 5] & e[1, 6] \\ e[2, 5] & e[2, 6] \\ e[3, 5] & e[3, 6] \end{pmatrix} \\ \begin{pmatrix} e[2, 1] & e[2, 2] \\ e[3, 1] & e[3, 2] \\ e[4, 1] & e[4, 2] \end{pmatrix} & \begin{pmatrix} e[2, 2] & e[2, 3] \\ e[3, 2] & e[3, 3] \\ e[4, 2] & e[4, 3] \end{pmatrix} & \begin{pmatrix} e[2, 3] & e[2, 4] \\ e[3, 3] & e[3, 4] \\ e[4, 3] & e[4, 4] \end{pmatrix} & \begin{pmatrix} e[2, 4] & e[2, 5] \\ e[3, 4] & e[3, 5] \\ e[4, 4] & e[4, 5] \end{pmatrix} & \begin{pmatrix} e[2, 5] & e[2, 6] \\ e[3, 5] & e[3, 6] \\ e[4, 5] & e[4, 6] \end{pmatrix} \\ \begin{pmatrix} e[3, 1] & e[3, 2] \\ e[4, 1] & e[4, 2] \\ e[5, 1] & e[5, 2] \end{pmatrix} & \begin{pmatrix} e[3, 2] & e[3, 3] \\ e[4, 2] & e[4, 3] \\ e[5, 2] & e[5, 3] \end{pmatrix} & \begin{pmatrix} e[3, 3] & e[3, 4] \\ e[4, 3] & e[4, 4] \\ e[5, 3] & e[5, 4] \end{pmatrix} & \begin{pmatrix} e[3, 4] & e[3, 5] \\ e[4, 4] & e[4, 5] \\ e[5, 4] & e[5, 5] \end{pmatrix} & \begin{pmatrix} e[3, 5] & e[3, 6] \\ e[4, 5] & e[4, 6] \\ e[5, 5] & e[5, 6] \end{pmatrix} \\ \begin{pmatrix} e[4, 1] & e[4, 2] \\ e[5, 1] & e[5, 2] \\ e[1, 1] & e[1, 2] \end{pmatrix} & \begin{pmatrix} e[4, 2] & e[4, 3] \\ e[5, 2] & e[5, 3] \\ e[1, 2] & e[1, 3] \end{pmatrix} & \begin{pmatrix} e[4, 3] & e[4, 4] \\ e[5, 3] & e[5, 4] \\ e[1, 3] & e[1, 4] \end{pmatrix} & \begin{pmatrix} e[4, 4] & e[4, 5] \\ e[5, 4] & e[5, 5] \\ e[1, 4] & e[1, 5] \end{pmatrix} & \begin{pmatrix} e[4, 5] & e[4, 6] \\ e[5, 5] & e[5, 6] \\ e[1, 5] & e[1, 6] \end{pmatrix} \end{pmatrix}$$

A typical block might look like the following.

$$\begin{pmatrix} e[2, 4] & e[2, 5] \\ e[3, 4] & e[3, 5] \\ e[4, 4] & e[4, 5] \end{pmatrix}$$

For each of the $\sigma \times (p - 1)$ blocks, we now create rules. Some of these rules may be “position independent”, that is, constant for all blocks depending only on the value of the entire automaton. Other rules may be “position dependent” and depend on the content of the block involved or the position of the block within the partitioned array. We then select a particular expression within the block—the one lying in the second row and first column—to transform to a fixed point using these rules via *Mathematica*’s **ReplaceRepeated** construct. We then make that fixed point of the transformation an argument to a function **combo** that takes the entire *untransformed* block as its second argument. Thus, through partitioning of the original array of programs and the **combo** function, the programs transform themselves. The second column of programs in each block becomes data to the transformed version of the program in the second row and first column of each block. [Note13]

This functional method is used over the entire partitioned set of expressions using the **MapIndexed** construct that permits each block to be aware of its position in the partitioned landscape. This methodology permits each program to evolve according to its neighborhood of “surrounding” programs. The following Code Fragment 15 shows the *Mathematica* code needed to accomplish this complex procedure and the structure of the end product of this method. [Note14]

```
illustrateEvolution =
  Shallow[MatrixForm[evolvedprograms =
    MapIndexed[
      combo[ReplaceRepeated[#1[[2, 1]],
        somerules[globalproperties[a, #1, #2]]], #1] &,
      partitionedabstractarray, {2}]], 6] (15)
```

combo[	combo[	combo[	combo[	combo[
ReplaceRepeated[	ReplaceRepeated[	ReplaceRepeated[	ReplaceRepeated[	ReplaceRepeated[
<<2>>], {<<3>>}]	<<2>>], {<<3>>}]	<<2>>], {<<3>>}]	<<2>>], {<<3>>}]	<<2>>], {<<3>>}]
combo[	combo[	combo[	combo[	combo[
ReplaceRepeated[	ReplaceRepeated[	ReplaceRepeated[	ReplaceRepeated[	ReplaceRepeated[
<<2>>], {<<3>>}]	<<2>>], {<<3>>}]	<<2>>], {<<3>>}]	<<2>>], {<<3>>}]	<<2>>], {<<3>>}]
combo[	combo[	combo[	combo[	combo[
ReplaceRepeated[	ReplaceRepeated[	ReplaceRepeated[	ReplaceRepeated[	ReplaceRepeated[
<<2>>], {<<3>>}]	<<2>>], {<<3>>}]	<<2>>], {<<3>>}]	<<2>>], {<<3>>}]	<<2>>], {<<3>>}]
combo[	combo[	combo[	combo[	combo[
ReplaceRepeated[	ReplaceRepeated[	ReplaceRepeated[	ReplaceRepeated[	ReplaceRepeated[
<<2>>], {<<3>>}]	<<2>>], {<<3>>}]	<<2>>], {<<3>>}]	<<2>>], {<<3>>}]	<<2>>], {<<3>>}]
combo[	combo[	combo[	combo[	combo[
ReplaceRepeated[	ReplaceRepeated[	ReplaceRepeated[	ReplaceRepeated[	ReplaceRepeated[
<<2>>], {<<3>>}]	<<2>>], {<<3>>}]	<<2>>], {<<3>>}]	<<2>>], {<<3>>}]	<<2>>], {<<3>>}]

Zooming in, we find that the third site’s new second expression is as follows.

```
combo[e[3, 2] //. somerules[globalproperties[a, {{e[2, 2], e[2, 3]},
  {e[3, 2], e[3, 3]}, {e[4, 2], e[4, 3]}}, {3, 2}]],
  {{e[2, 2], e[2, 3]}, {e[3, 2], e[3, 3]}, {e[4, 2], e[4, 3]}}}
```

And the fourth site's fifth expression is the following.

```
combo[e[4, 5] /. somerules[globalproperties[a, {{e[3, 5], e[3, 6]},
  {e[4, 5], e[4, 6]}, {e[5, 5], e[5, 6]}}, {4, 5}]],
  {{e[3, 5], e[3, 6]}, {e[4, 5], e[4, 6]}, {e[5, 5], e[5, 6]}}]
```

We generalize this portion of the process for use on any automaton by defining a function **update** as follows. [Note15]

```
update[a : automatonPattern,
  positionindependenttransformationrules_,
  positiondependenttransformationrules_] :=
MapIndexed[
  combo[ReplaceRepeated[#1[[2, 1]],
    Join[positionindependenttransformationrules,
      positiondependenttransformationrules[#1, #2]], #1] &,
    Partition[getRing[a], {3, 2}, {1, 1}, {{2, 1}, {-2, 2}}, {2}]]] (16)
```

The last column in this array that results from the output of **update** is of particular interest since it represents the new intended behaviors of each site. Using these intended behaviors, we can determine actual levels of production (after cow trampling is taken into account). From actual levels of production, we can determine equilibrium prices, and from equilibrium prices, we can determine each site's immediate and cumulative profitability.

Finally, there is the possibility of exogenous mutations by dropping sites with too low levels of profitability or cumulative profitability and replacing them by sites with randomly chosen characteristics. The function **killandreplace** shown below in Code Fragment 17 implements that procedure. [Note16]

```
Options[killandreplace] =
  {verbose → False, veryverbose → False, mreport → False};
SetAttributes[killandreplace, HoldAll];
killandreplace[dataselectionfunction_, deathcriteria_,
  replacementgenerator_, opts___][a : automatonPattern] :=
setRing[a, Module[{thering, thedata, deathpos, replacements,},
  thedata = Map[dataselectionfunction, r];
  deathpos = Position[thedata, _?(deathcriteria), {1},
    Heads → False];
  replacements = Table[replacementgenerator[],
    {Length[deathpos]}];
  If[verbose /. {opts} /. Options[killandreplace],
    Print["killing ", deathpos];];
  If[veryverbose /. {opts} /. Options[killandreplace],
    Print[#[[6]] & /@ replacements];];
  If[mreport /. {opts} /. Options[killandreplace],
    AppendTo[mutationreport, deathpos];];
  If[deathpos === {}, r, ReplacePart[r, replacements,
    deathpos, Partition[Range[Length[deathpos]], 1]]]]] (17)
```

We thus end up with an evolved automaton and begin the process all over again.

The code for **process** thus looks as follows.

```

combo[x_Function, y_] := x[y];
combo[x_, _] := x
(18)

process[costfunction_, compensationfunction_, damagefunction_,
  computationcostfunction_, dataselectionfunction_,
  deathcriteria_, replacementgenerator_, cowtramplefactor_,
  opts___][a : automatonPattern] :=
Module[{myagd, pl, stage1, stage2, stage3, stage4, stage5,
  actualintensities, newprices, temp, theglobals},
  pl = profitList[a];
  myagd = {localbiggest[pl], localmedian[pl], localsmallest[pl],
    intendedIntensityList[a], Length[pl]};
  temp = update[a, pier[a], pder[myagd]];
  stage1 =
    automaton[
      Transpose[Join[Take[Transpose[r], 1], Transpose[temp]]],
      globals];
  If[Not[MatchQ[stage1, automatonPattern]],
    Print["Stage 1 Error", stage1]];
  stage2 = setActualIntensityList[stage1,
    computeActualIntensities[cowtramplefactor][
      behaviorList[stage1]]];
  If[Not[MatchQ[stage2, automatonPattern]],
    Print["Stage 2 Error", stage2]];
  stage3 = setPrices[stage2,
    computePrices[inverseDemandFunctions /. {globals},
      totalActualIntensities[stage2]]];
  If[Not[MatchQ[stage3, automatonPattern]],
    Print["Stage 3Error", stage3]];
  theprofits =
    computeProfits[costfunction, compensationfunction,
      damagefunction, computationcostfunction][stage3];
  stage4 = setProfitList[stage3, theprofits];
  If[Not[MatchQ[stage4, automatonPattern]],
    Print["Stage 4 Error", stage4]];
  stage5 = setCumulativeProfitList[stage4,
    cumulativeProfitList[a] + theprofits];
  If[Not[MatchQ[stage5, automatonPattern]],
    Print["Stage 5 Error", stage5]];
  killandreplace[dataselectionfunction, deathcriteria,
    replacementgenerator, opts][stage5]
]
(19)

```

Evolution is then modeled as the repeated application of some variant of a process function for the earlier of a finite number of iterations or until the application yields an expression not matching an **automatonPattern**. Code Fragment 20 shows how *Mathematica* can do this in a way that spits out just the end result of this process. Code Fragment 21 shows how *Mathematica* can do this in a way that spits out all iterations of this process. And Code Fragment 22 shows how *Mathematica* can do this in a way that spits out every n th iteration of this process in which the user designates n .

```

evolution[pfunc_][ic : automatonPattern, iters_] :=
  NestWhile[pfunc, ic, Function[MatchQ[#, automatonPattern]],
    1, iters]

```

(20)

```

evolutionList[pfunc_][ic : automatonPattern, iters_] :=
  NestWhileList[pfunc, ic, Function[MatchQ[#, automatonPattern]],
    1, iters]

```

(21)

```

evolutionSkipList[pfunc_][ic : automatonPattern, inneriters_,
  outeriters_] :=
  evolutionList[evolution[pfunc][#, inneriters] &][ic, outeriters]

```

(22)

■ Conclusion

The techniques and code outlined in this article can be put to many uses. The samples below illustrate how these techniques have been used to examine a standard problem in the economic analysis of law. Hitherto, this problem has generally been studied using only conventional techniques based on the dubious assumptions of complete rationality and optimization. We have used our work to study the evolution of behavior and learning in an “economy” in which the costs of the activities of the agents are sometimes externalized onto neighboring agents. We find that, under certain circumstances, including certain rules of law, the behavior of the agents coevolves to resemble closely that which an external authority might impose in order to maximize wealth. A fuller understanding of these experiments can be found in the notebooks listed under Additional Materials, available through *The Mathematica Journal* web site. We hope that the samples below will stimulate an appetite for further examination of these notebooks and for further explorations in this field.

□ A Graphic Showing the Evolution of Behaviors in an Economy

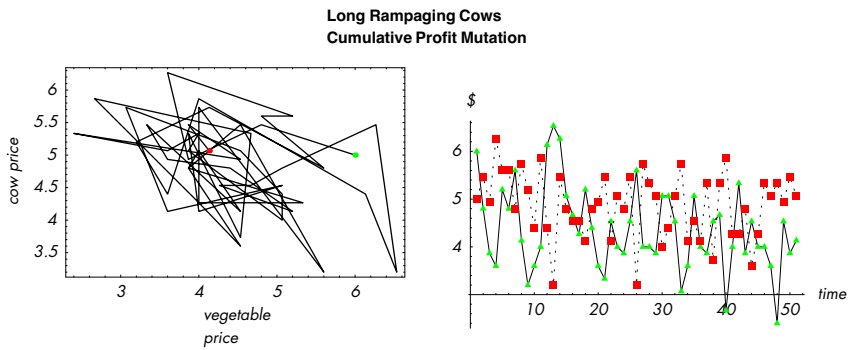
In the image below, the darker areas represent farming. The lighter areas represent ranching. As in Figure 2 above, each row represents the economy at one point in time. Time moves forward as one moves down the image.

Long Rampaging Cows
Cumulative Profit Mutation



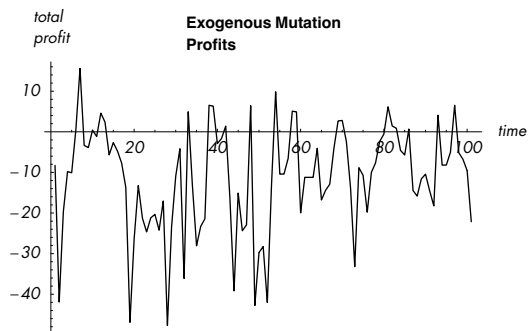
□ **A Graphics Array Showing Market Prices in an Evolving Economy**

This graphic illustrates how the prices of the two products generated by the sample economy (vegetables and cows) move together as the agents evolve their behaviors.



□ **A Graphic Showing Profits in an Evolving Economy**

This graphic illustrates how the total wealth of the economy varies as the agents evolve their behaviors.



□ A Table Showing Behavior Programs of Agents in an Economy After Co-Evolution

This table gives the behavior programs of the agents in an economy after coevolving under certain rules of law. These programs are generated through the process described in this article.

behavior[direction[if[boolean[threebitint[5] ≥ threebitint[5]], direction[medianprofitable], direction[self]]]]
behavior[{activity[farmer], threebitint[modplus[threebitint[3], threebitint[4]], 0, 3.8506, 0}]
behavior[direction[leastprofitable]]
behavior[direction[left]]
behavior[replacepart[behavior[direction[right]], behavior[oppositeactivity[behavior[direction[right]]]], {part[activitypart]}]]
behavior[oppositeactivity[behavior[{activity[rancher], threebitint[3], 0, 3.8506, 0}]]]
behavior[{activity[farmer], threebitint[0], 0, 3.8506, 0}]
behavior[replacepart[behavior[direction[medianprofitable]], behavior[direction[medianprofitable]], {part[intensitypart]}]]
behavior[{activity[farmer], threebitint[0], 0, 3.8506, 0}]
behavior[direction[leastprofitable]]
behavior[direction[right]]
behavior[{activity[rancher], threebitint[modtimes[-1, threebitint[rotateleft[threebitint[0], threebitint[1]]]]], 0, 3.8506, 0}]
behavior[oppositeactivity[behavior[direction[right]]]]
behavior[{activity[rancher], threebitint[1], 0, 3.8506, 0}]
behavior[direction[leastprofitable]]

□ **A Table Showing How Agents in an Economy Evolve to Learn How to Learn**

A feature of our experiment was to let the agents evolve their own rules for learning and, indeed, for learning how to learn. The table below provides an example of these meta-programs that coevolve in our economy.

recombination[direction[mostprofitable], thedirection, threebitint[6], threebitint[2]]
recombination[none]
recombination[direction[leastprofitable], athreebitint, threebitint[0], threebitint[1]]
recombination[none]
recombination[none]
recombination[direction[mostprofitable], thedirection, threebitint[4], threebitint[1]]
recombination[none]
recombination[none]
recombination[direction[mostprofitable], athreebitint, threebitint[5], threebitint[modtimes[-1, threebitint[1]]]]
recombination[none]
recombination[none]
recombination[none]
recombination[none]
recombination[none]
recombination[direction[mostprofitable], athreebitint, threebitint[0], threebitint[5]]

■ **Acknowledgment**

Professor Chandler gratefully acknowledges the financial support of the University of Houston Law Foundation in the preparation of this article.

■ **End Notes**

- [1] The pioneering work in cellular automata was done by Stephen Wolfram [8].
- [2] The sum mod 2 of any number is the remainder that is left when one divides the number by two. Thus 3 mod 2 is 1.
- [3] The neighborhoods are sorted in decreasing order of their representation as a base 2 numeral.
- [4] It takes 2^{30} because the site and its relevant neighbors can take on 2^{27} different values, and each output is eight binary digits long.
- [5] Sipper [9] gets around this problem through an approach known as cellular programming. We choose a different approach here.

- [6] As discussed in more detail below, it is also possible to specify “translation rules” that, when used over and over again against expressions written in the grammar, transform those expressions into ones that the *Mathematica* kernel is able to simplify. The patterns produced by a grammar thus become executable programs capable of yielding a result such as a number or other value.
- [7] When the price is 4, the amount logically supplied by an optimizing producer is 4 because the marginal cost at 4 is 4. Thus, total revenue will be 16, which is the price of 4 multiplied by the optimal quantity 4. But 16 is also the amount that it costs to produce 4 units. Hence, the producer is behaving optimally and making zero economic profit, which essentially defines equilibrium.
- [8] δ is some arbitrary number that represents the price that will drive demand to zero.
- [9] The utilities make extensive use of the names for patterns provided in the previously defined expressions `automatonPattern` and `sitePattern`.
- [10] Increasing the size of these computation costs coupled with various forms of evolution favoring the more profitable behavior programs should result in more parsimonious programs.
- [11] A functional programming style is one in which expressions that are capable of operating on other expressions are treated as “first class” data and used frequently as parts of surrounding expressions.
- [12] The array has six columns rather than five because an additional column is needed to hold the actual behavior.
- [13] The first program at each site remains constant, however.
- [14] The `combo` function avoids a tricky issue of infinite recursion. When the first argument of `combo` bears a head of `Function`, `combo` yields that function applied to the second argument. Otherwise, `combo` just yields the first argument. Permitting the first argument to have a `Function` head and various expressions with the head of `Slot` as its argument eliminates a problem of infinite recursion that would otherwise occur. Suppose, for example, that a process of transformation on the expression `recombination[none]` should yield a pointer to an expression `recombination[none]`. If we then go grab the pointer to `recombination[none]` we set off an infinite recursion because, as known, `recombination[none]` is subject to transformation. We thus terminate the transformation process by giving the pointer a head of `Slot` and wrapping the pointer in a head of `Function`. We make sure not to have any transformations that operate on patterns with a head of `Function`. With the transformation process now complete, we use `combo` to apply the created `Function` expression to the object pointed to and resolve the reference. This method works so long as “terminal expressions” such as `recombination[none]` cannot appear inside of other evaluable expressions. Thus, if the expression `behavior[2]` transformed to `behavior[2]`, the expression `behavior[behavior[2]]` would be extremely difficult to handle.
- [15] The transformation rules are separated into those that are position-independent and those that are position-dependent to minimize recomputation of rules for each of the $\sigma \times (p - 1)$ blocks the `MapIndexed` function goes over.
- [16] The `HoldAll` attribute must be used to prevent the replacement generation function from evaluating (once) before it is needed to create random expressions. Otherwise, all new sites would be the same.

■ References

- [1] R. Coase, "The Problem of Social Cost," *Journal of Law and Economics*, **3**, 1960 pp. 1–44.
- [2] S. Chandler, "Modeling Law: Using Cellular Automata to Study Legal Regulations of Conflicting Land Use," in *Innovation in Mathematics Proceedings of Second International Mathematics Symposium* (V. Keränen, P. Mitic & A. Hietamäki, eds.), Southampton: Computational Mechanics Publications, 1997 pp. 73–82.
- [3] J. R. Koza, *Genetic Programming: on the programming of computers by means of natural selection*, Cambridge, MA: MIT Press, 1992.
- [4] C. Jacob, "Genetic L-system Programming: Breeding and Evolving Artificial Flowers with *Mathematica*," in *Mathematics with Vision: Proceedings of the First International Mathematics Symposium* (V. Keränen, P. Mitic, eds.), Southampton: Computational Mechanics Publications, 1995 pp. 215–222.
- [5] C. Jacob, "Simulating Evolution with *Mathematica*," in *Innovation in Mathematics: Proceedings of the Second International Mathematics Symposium* (V. Keränen, P. Mitic & A. Hietamäki, eds.), Southampton: Computational Mechanics Publications, 1997 pp. 263–272.
- [6] C. Jacob, *Principia Evolvica: Simulierte Evolution mit Mathematica*, Heidelberg: Dpunkt-Verlag, 1997.
- [7] C. Jacob, *Illustrating Evolutionary Computation with Mathematica*, San Francisco: Morgan Kaufmann Publishers, 2001.
- [8] S. Wolfram, *Cellular Automata and Complexity: Collected Papers*, Reading, MA: Addison-Wesley, 1994.
- [9] M. Sipper, *Evolution of Parallel Cellular Machines: The Cellular Programming Approach*, Heidelberg: Springer, 1997.
- [10] T. Yamamoto, "Order and Disorder in 2-D Cellular Automata with Heterogeneous Rules," *Complexity International* [online], **3**, 1996. <http://www.csu.edu.au/ci>

■ Additional Material

Available at www.mathematica-journal.com.

□ *Mathematica* Notebooks

CAGPValidation.nb
 CAGPImplementation.nb
 GP-ExprGeneration.nb

□ Data Files

cagpdatafiles.zip

About the Authors

Professor Seth J. Chandler received his J.D. from the Harvard Law School (1983) and his A.B. from Princeton University (Woodrow Wilson School of Public and International Affairs) (1979). He is now an associate professor of law at the University of Houston Law Center. A pioneer in the use of *Mathematica* in the economic analysis of law, he continues his use of this computer system in studies on insurance law and "Foggy Game Theory".

Professor Christian Jacob received his Ph.D. in computer science from the University of Erlangen-Nuremberg, Germany (1995). He is now an assistant professor in the Department of Computer Science at the University of Calgary, Canada. He is the author of two textbooks on evolutionary computation with *Mathematica* [6, 7]. His current research focuses on recombinations of evolutionary and adaptive swarm intelligence systems.

Abstracts of Additional Papers from IMS '99

■ **Mathematica and Didactical Innovation**

G. Albano, C. D'Apice, S. Salerno

*Dipartimento di Ingegneria dell'Informazione e Matematica Applicata
Università degli Studi di Salerno
via Ponte don Melillo
84084 Fisciano, Italy*

A. Cavalloney

*Centro di Ricerca in Matematica Pura e Applicata
Università degli Studi di Salerno
via Ponte don Melillo
84084 Fisciano, Italy*

A package for education in mathematics that focuses on the study of a conic is presented. The package is structured in a hypertextual way, including a part on theory and a guided lab for the practical session. The most meaningful innovation is given by the verification routines. The user may solve exercises by hand and check the solutions he/she finds by using two verification functions.

south.rotol.ramk.fi/~keranen/IMS99/paper20/ims99paper20.pdf

■ An Algorithm Computing Homology Groups of Commutative Differential Graded Algebras with Linear Differential

V. Alvarez, J.A. Armario, M.D. Frau, P. Real, B. Silva

Departamento de Matemática Aplicada I, Facultad de Informática y Estadística

Universidad de Sevilla

Avda. Reina Mercedes s/n

41012 Sevilla, Spain

valvarez@cica.es; armario@cica.es; real@cica.es; silva@cica.es and lfrau@euler.fie.us.es

We present here an implementation in *Mathematica* of an algorithm for computing the integer algebra homology of commutative differential graded algebras with linear differential. This algorithm is essentially based on the method described in V. Alvarez, J. A. Armario, P. Real, and B. Silva, "HPT and computability of the homology of commutative DGA-algebras," Conference on Secondary Calculus and Cohomological Physics, Moscow, August, 1997.

south.rotol.ramk.fi/~keranen/IMS99/paper9/ims99paper9.pdf

■ Normal Lines Drawn to Ellipses and Elliptic Integrals

Tilak de Alwis

Department of Mathematics

Southeastern Louisiana University

Hammond, LA 70402, USA

FMAT1117@SELU.EDU

Consider the ellipse in the standard position, and P , an arbitrary point on the plane. Finding the minimum distance from P to the ellipse is a well-known problem. One can easily show that the minimum distance path lies along a normal line to the ellipse, passing through P . This paper deals with a study of all the normal lines drawn from point P to the ellipse. We have used *Mathematica* to illustrate and discover several aspects of these normal lines. *Mathematica* can be used in contemporary mathematical research and education as a computational, visualization, experimentation and conjecture-forming tool. The paper well illustrates such usage via a study of the normal lines to the ellipse. We used *Mathematica* 3.0 on a Windows 95 platform.

south.rotol.ramk.fi/~keranen/IMS99/paper54/ims99paper54.pdf

south.rotol.ramk.fi/~keranen/IMS99/paper54/ims99paper54.nb

■ Integration of *Mathematica* into an Information System Supporting Chemical Process Development

Juha R. Anttila

*Laboratory of Chemical Process Engineering
Department of Process Engineering
University of Oulu, Finland*

Antti P. Pasanen

*Technical Research Centre of Finland
Chemical Technology*

Mathematica has been used as an environment for object modeling a new chemical-pulping process. The usability of *Mathematica* has been enhanced by integrating it into an Information System (IS) supporting chemical process development. The implementation of IS is based on a new phenomenon-driven process-design methodology. The integration into the IS results in an easy way to change and give process structures to *Mathematica*, and makes *Mathematica* an attractive alternative compared to commercial simulators.

south.rotol.ramk.fi/~keranen/IMS99/paper19/juttu.pdf
south.rotol.ramk.fi/~keranen/IMS99/paper19/Juttu.doc

■ Symbolic Computations in Problems of Mechanics

Andrej V. Banshchikov, Larissa A. Bourlakova, Galina N. Ivanova, Valentin D. Irtegov, Tatyana N. Titorenko

*Institute of System Dynamics and Control Theory
Siberian Branch Russian Academy of Sciences
134, Lermontov str., Irkutsk, 664033, Russia
irteg@icc.ru*

This report considers the long-term research on development of algorithms and automation of complicated mechanical and controlled systems. We have created specialized systems of computer algebra (CAS) and software packages “Dynamics”, “Mechanics”, the possibilities and the algorithms of which are described in [1–5] and others. Currently we are developing software systems [6–9] for solving mechanical problems based on CAS *Mathematica* [10]. The software allows us to automate, and consequently, speed up processes of the modeling and dynamic analysis of complicated systems, and to avoid errors at all stages of research. The base of the algorithms, which are realized in these packages, was formed by

classical methods of analytical mechanics and stability theory. Our experience with symbolic computation packages allows us to conclude that CAS are tools for research in the field of theoretical mechanics.

south.rotol.ramk.fi/~keranen/IMS99/paper15/ims99paper15.pdf

■ The Structuring Power of *Mathematica* in Mathematics and Mathematical Education

R. Barrere

*University of Franche-Comte
ENSM 26 chemin de l'Épitaphe
F-25000 Besançon, France
rbarrere@ens2m.fr*

So far, *Mathematica* has been used mainly either in teaching as a support for classical courses, or in research as a computing environment. However, it could be assigned a wider and more profound role in mathematics and mathematical education. In particular, it could support, facilitate and accelerate the integration of algorithmic aspects into the corpus of mathematics.

south.rotol.ramk.fi/~keranen/IMS99/paper35/ims99paper35.pdf
south.rotol.ramk.fi/~keranen/IMS99/paper35/barrere.nb

■ Fractional Calculus: Application and Results

Gerd Baumann

*Department of Mathematical Physics
University of Ulm
Albert-Einstein-Allee 11
D-89069 Ulm, Germany*

Over the last two decades, anomalous relaxation and diffusion processes have been studied from both the experimental and theoretical point of view. Anomalous transport processes occur usually in so-called complex (that is, disordered) systems. Here, we review some recent developments in modeling non-standard relaxation and diffusion equations based on Riemann–Liouville fractional calculus techniques. Closed analytical expressions of the solutions of such fractional differential equations are given in terms of Fox's H-functions. The analytical calculations carried out are fully supported by our *Mathematica* package **FractionalCalculus**.

south.rotol.ramk.fi/~keranen/IMS99/paper46/FractionalCalculus.nb

■ Sound as an Additional Learning Tool

Johan Berglind

*Chalmers University College
Göteborg, Sweden*

One of the advantages of using *Mathematica* in basic math courses is that it opens up the possibility of creating experiments in mathematics. In electrical engineering courses at Chalmers University College, we have used sound as an experimental tool to investigate aspects of mathematics that otherwise tend to elude students. We give examples from two areas: ordinary differential equations and Fourier analysis. In the first example, we let *Mathematica* solve a differential equation with a parameter and then play the sound of the solution for different values of that parameter. The object is to find beat and resonance by listening. In the second example, we expand different functions in the Fourier series and then listen to the functions and partial sums of these series.

south.rotol.ramk.fi/~keranen/IMS99/paper30/paper_30patch_for_IMS99.nb

■ PLEMATH SYSTEM: Linear and Integer Linear Programming in the *Mathematica* System

Mijail Borges-Quintana, Miguel A. Borges-Trenard

*Department of Mathematics
Faculty of Sciences
University of Oriente
Santiago de Cuba, 90500, Cuba
fmijail@csd.uo.edu.cu; mborgesg@csd.uo.edu.cu*

Edgar Martínez-Moro

*Applied Mathematics Department
E.T.S. Arquitectura University of Valladolid
Valladolid, 47014, Spain
edgar@vax631.cpd.uva.es*

The PLEMATH System has been built as an educational tool for Linear Programming and Integer Linear Programming. In this work we show examples that illustrate the use of the system. We describe how the commands executing the different algorithms incorporated into PLEMATH have been made, as well as their possibilities and limitations. Moreover, we exhibit some ideas for further extensions of this work. No previous knowledge of *Mathematica* is supposed.

south.rotol.ramk.fi/~keranen/IMS99/paper16/plemath.pdf
south.rotol.ramk.fi/~keranen/IMS99/paper16/Course.nb

■ Pseudospectral Symbolic Computation for Financial Models

F.O. Bunnin, Y. Ren, Y. Guo, J. Darlington

Department of Computing / Fujitsu Parallel Computing Research Centre

Imperial College

London, SW7 2 BZ, UK

fobl@doc.ic.ac.uk; yr@doc.ic.ac.uk; yg@doc.ic.ac.uk; jd@doc.ic.ac.uk

The modeling of financial markets as continuous stochastic processes provides the means to analyze the implications of models and to compute prices for a host of financial instruments. We code, as a symbolic computing program, the analysis, initiated by Black, Scholes and Merton, of the formation of a partial differential equation whose solution is the value of a derivative security, from the specification of an underlying security's process. The Pseudospectral method is a high-order solution method for partial differential equations that approximates the solution by global basis functions. We apply symbolic transformations and approximating rewrite rules to extract essential information for the Pseudospectral Chebyshev solution. We write these programs in *Mathematica*. Our C++ template implementing general solver code is parameterized with this information to create instrument and model-specific pricing code. The Black–Scholes model and the Cox–Ingersoll–Ross term-structure model are used as examples.

south.rotol.ramk.fi/~keranen/IMS99/paper45/ims99paper45.pdf

■ System Stochasticity: Discrete Formulation with *Mathematica*

Gautam Dasgupta

Professor, Civil Engineering and Engineering Mechanics

Columbia University

600 Seeley W. Mudd

New York, NY 10027, USA

gd18@columbia.edu

Material and geometrical randomness are depicted by the variability of numerical solutions. This second-order effect in response cannot be faithfully captured unless the equilibrium equation is exactly satisfied. Hence engineering approximations of system stochasticity demand higher accuracy than their deterministic counterparts.

Consequently, the contamination in numerical responses cannot be removed by selecting large Monte Carlo samples. Stochastic shape functions and stochastic Green's functions constitute the bases for finite and boundary elements, respectively. These fundamental solutions need to be modified distinctly for each

Monte Carlo representative via symbolic manipulation of the governing algebraic equation. The subsequent closed-form analytical integration of the energy density function is illustrated here for a beam problem with geometrical stochasticity after zero shear locking is met in a patch test.

south.rotol.ramk.fi/~keranen/IMS99/paper56/ims99paper56.nb

■ Computing of Curvature Invariants in Arbitrary Dimension

Stanislaw Ewert-Krzemieniewski

Technical University of Szczecin

The *MathTensor* software enables one to calculate the curvature invariants of differentiable manifolds in the case that the dimension of manifold under consideration is a concrete number. When proving the existence of geometric objects it is important to have examples for all dimensions n , greater or equal to some n_0 , that is, when the dimension is just a symbol. The aim of this paper is to show how, using *MathTensor* and *Mathematica* commands, one can define the covariant and contravariant components of the metric tensor. Then we will show how to define different curvature tensors and how to execute the calculation of their components.

south.rotol.ramk.fi/~keranen/IMS99/paper53/ims99paper53.pdf

■ Nontrivial Asymptotic Formulas by Symbolic Computation

Giuliano Gargiulo, Saverio Salerno

We want to study the asymptotic behavior of the complete elliptic integral of the first kind $K(m)$ when $m \rightarrow 1$. This is motivated, for example, by the occurrence of $K(m)$ as capacitance of a circular capacitor with slit (or similar geometries—see for example [9]), m being essentially the ratio between the slit's length and the radius of the circle. We show that the analysis of the asymptotic behavior can be done in several ways (including series expansion and summation, symbolic integration and computation of limits) using both the numerical and symbolical capabilities of state-of-the-art computer algebra systems.

south.rotol.ramk.fi/~keranen/IMS99/paper21/ims99paper21.nb

■ Dynamic Optimization and Differential Games with Applications to Economics

Yuji Itaya

Ph.D. in Engineering, Professor

Department of Information Management

Asahi University

Hozumi, Motosu, Gifu, 501-0296, Japan

itaya@alice.asahi-u.ac.jp

At IMS '97 we showed several packages for:

- (1) solving simultaneous equations in real domain;
- (2) obtaining necessary conditions of constrained static optimization problems;
- (3) solving simultaneous nonlinear equations approximately;
- (4) determining signs of expressions;
- (5) solving differential equations approximately;
- (6) solving boundary-value problems approximately; and
- (7) solving dynamic optimization problems approximately.

We also presented applications of these packages to urban economic models [1]. However, applications are limited in the package for solving dynamic optimization problems, because there exist many problems to which the package cannot be applied. In this paper, we revise and expand it so that it can be widely used. By further expanding the package above, we can solve differential games. We first show a solution of simple differential games with *Mathematica* and view the optimal trajectories. We then present packages for differential games by upgrading the dynamic optimization package and show some applications to economic problems.

south.rotol.ramk.fi/~keranen/IMS99/paper27/DifGame.pdf

south.rotol.ramk.fi/~keranen/IMS99/paper27/IMS99_No27.nb

■ The Visualization of Different Fiber-Optic Loop Interferometer Applications Work by Applying 3D *Mathematica* Graphics

Leszek R. Jaroszewicz

Institute of Applied Physics

Military University of Technology

Kaliskiego 2

00-908 Warsaw 49, Poland

jarosz@wat.waw.pl or jarosz@jkkeds.com.pl

The usefulness of the *Mathematica* 3D-visualization and animation for investigation of fiber-optic loop interferometers in different engineering applications

work is presented. The Jones' matrix calculus has been used for description of fiber-optic elements of all interferometric systems. Such uniform description gives a possibility to obtain analytical relations, which depend on fundamental system parameters. Unfortunately, these relations are usually too complicated for physical interpretation. The possibility of their analysis by the animated 3D graphics is also shown in the present paper.

south.rotol.ramk.fi/~keranen/IMS99/paper26/ims99paper26.pdf

■ **MathModelica: A New Modeling and Simulation Environment for *Mathematica***

Mats Jirstrand, Johan Gunnarsson

*MathCore AB
Wallenbergs gata 4
SE-583 35 Linköping, Sweden*

Peter Fritzson

*PELAB (Programming Environment Lab)
Department of Computer and Information Science
Linköping University
SE-581 83 Linköping, Sweden*

MathModelica is a *Mathematica* extension, which provides a modeling and simulation environment for *Mathematica* based on the new standard of physical modeling languages called “Modelica”. Modelica is a new object-oriented multi-domain modeling language based on algebraic and differential equations. In this paper we present a language and an environment, MathModelica, that integrates different phases of the Modelica development life-cycle. This is achieved by using the *Mathematica* environment and its structured documents, that is, notebooks. Simulation models are represented in the form of structured documents, which integrate source code, documentation and code transformation specifications, as well as providing control over simulation and result visualization. Import and export of Modelica code between internal structured and external textual representation is supported. *Mathematica* is an interpreted language, which is suitable as a scripting language for controlling simulation and visualization. *Mathematica* also supports symbolic transformations on equations and algebraic expressions which is useful in building mathematical models.

south.rotol.ramk.fi/~keranen/IMS99/paper12/MathModelica-IMS99.nb

■ **Mathematica as a Communication-Enhancement Tool in the Classroom**

Hiroshi Kimura

Kyushu Institute of Technology, Japan

Chikara Miyaji

Tsukuba University, Japan

Shinya Ohashi

Kashiwa High School, Japan

Kenji Yoshida

Konan Boys' High School, Japan

To use computers in math classes, it is required that the computers can do math well and are easy to use. Moreover, if they help communication among students and between students and teachers, they enlarge the possibilities of interactions in the classroom. However, network software technology for education has not yet matured. Existing network software is sometimes powerless, being generally difficult to modify and hard to integrate with other educational software. In this presentation, we demonstrate the network tools that were built on *Mathematica*. Our goals are to provide tools that:

- 1) help various types of communications such as one-to-all, one-to-several, and one-to-one
- 2) help communications using not only text, but also various formats: structured text, mathematical formula, and pictures
- 3) can be used easily by the student
- 4) can be easily customized by teachers for their special needs

We report the effectiveness of the tools from our experiences.

south.rotol.ramk.fi/~keranen/IMS99/paper32/ims99paper32.pdf

south.rotol.ramk.fi/~keranen/IMS99/paper32/ims99paper32.nb

■ An Example of Computer-Based Study Material in Mathematics: Planning Principles and Realization

Simo K. Kivela

Institute of Mathematics

Helsinki University of Technology

Finland

Simo.Kivela@hut.fi

The development of software and network technology has created enormous possibilities for teaching and learning. At least it is widely believed that there is great potential. In practice, it is not so clear how these possibilities should be used. In this paper, an experiment in the field of mathematics at the university level is reported with respect to philosophy, planning principles, realizations, and future. The core of the experiment is a study package based on hypertext, web browsers, and computer algebra systems.

south.rotol.ramk.fi/~keranen/IMS99/paper52/IMS99kivela.pdf

■ Numerical Simulations of the Cubic Schrödinger Equations for Periodic Perturbations

Robert Knapp

Wolfram Research, Inc.

In the course of investigating the effect of a random potential on pulses with the cubic nonlinear Schrödinger equation, I became intrigued by what one should define as the effective width of a pulse or soliton, particularly in cases that are near-integrable. In an attempt to answer that question, I did some numerical simulations with periodic potentials and came across an interesting range of behavior depending on how the length scales of potential and pulse compare. Some of the results are explainable by simple perturbation methods, and some remain more mysterious. I show the results obtained and conclude with some observations about the effective width. I first give a brief description of the numerical method. Next, I present a leading order perturbation analysis for a slowly varying potential. Then, I show how well this theory bears out in numerical simulations with a slowly varying potential. The results get more interesting when the assumption that the potential is slowly varying breaks down. Beyond this, some simulations with a rapidly varying potential show some mildly surprising behavior. Finally, I try to bridge the entire range of length scales with some conclusions and questions.

south.rotol.ramk.fi/~keranen/IMS99/paper5/ims99paper5.html

■ **Mathematica Numerics**

Robert Knapp

Wolfram Research, Inc.

Selected examples of numerical computations using *Mathematica* 4.

south.rotol.ramk.fi/~keranen/IMS99/knapp/

■ **Transformation of Logical Specification into IP-formulas**

Qiang Li, Tetsuo Ida

Institute of Information Sciences and Electronics

University of Tsukuba, Japan

liq@score.is.tsukuba.ac.jp; ida@score.is.tsukuba.ac.jp

Yike Guo

Department of Computing

Imperial College, London, UK

yg@doc.ic.ac.uk

The classical algebraic modeling approach for integer programming (IP) is not suitable for some real-world IP problems, since the algebraic formulations allow only for the description of mathematical relations, not logical relations. In this paper, we present a language L+ for IP, in which we write a logical specification of an IP problem. L+ is a language based on predicate logic but extended with meta predicates such as “at least” (m, S), where m is a non-negative integer, meaning that at least m predicates in the set S of formulas hold. The meta predicates are introduced to facilitate reasoning about a model of an IP problem rigorously and logically. Using *Mathematica*, we can represent the logical formulas, called “L+ formulas”, efficiently and completely, and we can define a set of transformation rules and transform L+ formulas into IP formulas, finally simplifying the IP formulas in *Mathematica*. Also, by using *MathLink* and CGI programming, we develop a web-based interface to support the system with modeling language, transformation of IP, and IP solvers. This provides a web-based client-server model, in which the power of high-level IP modeling and high-performance IP solving can be integrated and developed for a wide range of business users to solve large-scale decision-making problems. The primary experiment indicates that *Mathematica* is a powerful tool for representing logical formulas and for the transformation of formulas and that *MathLink* is convenient for connecting *Mathematica* to other software platforms.

south.rotol.ramk.fi/~keranen/IMS99/paper23/ims99paper23.pdf

■ Testing Chaos and Fractal Properties in Economic Time Series

Maria I. Loffredo

Dipartimento di Matematica

Università di Siena

I-53100 Siena, Italy

loffredo@unisi.it

A search for empirical evidence of chaos and testing fractal and other statistical properties in the framework of time series analysis is carried out as a preparatory step in order to apply these concepts to the data proper of financial markets and deal with the puzzling failure of traditional economic theories. Concepts like correlation dimension and Lyapunov exponents are discussed and simple *Mathematica* programs are given for their evaluation. Before their application to real economic data, a test on a well-known nonlinear dynamical system, through the correspondent reconstructed phase space and time series, is carried out.

south.rotol.ramk.fi/~keranen/IMS99/paper25/ims99paper25.pdf
south.rotol.ramk.fi/~keranen/IMS99/paper25/ims99paper25.nb

■ Interactive Graphics Using *MathLink*

Chikara Miyaji

University of Tsukuba, Japan

Interactive graphics effectively expresses the dynamic structure of a model, and helps improve the intuitive understanding of concepts in the sciences. Also it can be used as a user-interface tool. Currently, *Mathematica* graphics is not interactive but static and merely a side effect of computation. It is not possible to select a part of a graphic using the mouse, nor can one make interactive dynamic relationships between graphical elements. The graphics system introduced here is a *MathLink* program for creating interactive graphical objects which the user can manipulate. In this program, all graphics elements—point, line, curve, and text—are represented as objects which can be manipulated by mouse; all graphics are manipulated by a kernel evaluation; and the relationship between objects can be defined as *Mathematica* functions. This paper outlines the system and discusses its design and implementation.

south.rotol.ramk.fi/~keranen/IMS99/paper24/ims99paper24.pdf
south.rotol.ramk.fi/~keranen/IMS99/paper24/ims99paper24.nb

■ **Mathematica-Aided Education of Science-Major Students**

K. Nakagami, F. Takeuchi, F. Ushitaki, M. Yasugi

Faculty of Science

Kyoto Sangyo University

Kyoto, Japan

www.kyoto-su.ac.jp/~yasugi

We make a comprehensive report of our experiences with *Mathematica*-aided education. Some students belonging either to the mathematics or computer science department do graduation works using *Mathematica* as a language. We take up three subjects in those works.

south.rotol.ramk.fi/~keranen/IMS99/paper31/ims99paper31.pdf

south.rotol.ramk.fi/~keranen/IMS99/paper31/ims99paper31.nb

■ **Interactive Online Exercises of Basic Mathematical Functions**

H. Nishizawa

Department of Electrical Engineering

Toyota National College of Technology

Eisei-cho 2-1

Toyota 471-8525, Japan

nisizawa@tctcc.cc.toyota-ct.ac.jp

In this paper, a system is described for serving online exercises of mathematical functions for high-school students. The system uses web browsers running on students' computers for displaying questions, answers, and explanations. Although recent developments of electrical worksheets and symbolic computing programs like *Mathematica* make it easy and quick for a student to draw graphs, it is only half the way to understanding. Because the drawing is automated and includes minimal guesswork, the student does not necessarily get ideas about, for example, the effect of a coefficient value on the shape or the position of a graph. The only rigid knowledge he has is how he operates the program. The online exercises provide the other half. The first thing a student does in the exercise is to make a guess trying to describe the given graph correctly. There are no instructions for expressing the graph, nor hints before the first guess. After the first guess is done, then comes the hint or comment which helps him to make the next guess. This makes a kind of experiment on the desktop. A series of experiments tells him the behavior of graphs in detail and the relation to expressions. Adding to that, because the evaluations are done symbolically using *Mathematica* functions, wide varieties of expressions are allowed, which is easier to accept for the average student. The exercise provides the students with opportunities to

learn from their mistakes and to build their own theory about mathematical functions.

south.rotol.ramk.fi/~keranen/IMS99/paper18/ims99paper18.pdf

■ ***Mathematica*-Based Package for Studying Ordinary Differential Equations and for Analyzing the Learning Process**

Riikka Nurmiainen

*Institute of Mathematics
Helsinki University of Technology
Finland
Riikka.Nurmiainen@hut.fi*

The design of a *Mathematica*-based learning environment for ordinary differential equations (ODE) is discussed. The ODE package consists of hypermedia-based information about differential equations and a collection of different kinds of exercises. The purpose is also to collect data about the student's work and behavior during the computer sessions using a recording system created as a part of the package. The data is used to analyze the learning process. The final goal is to get more knowledge about appropriate ways to use computer algebra systems like *Mathematica* in mathematics studies.

south.rotol.ramk.fi/~keranen/IMS99/paper51/ims99paper51.pdf

■ **On “*Mathematica* Natsu no Gakkou” (*Mathematica* Summer School)**

Shinya Ohashi

Kashiwa Municipal High School, Japan

Kenji Yoshida

Konan Boys' High School, Japan

Mathematica is well known among secondary school teachers in Japan. But not many of them have used *Mathematica* in the classroom or have developed teaching materials that use it. “*Mathematica* Natsu no Gakkou” is a project for providing the opportunity to develop *Mathematica*-related materials for teachers who do not have enough information on how to use *Mathematica*. The first *Mathematica* Natsu no Gakkou was held in summer 1996 with the cooperation of

Wolfram Research Asia Limited, the *Mathematica* User's Group in Japan, and *Mathematica* dealers in Japan. So far, about 60 teachers have attended these workshops and various teaching materials have been developed. Anyone can download and use them freely. This, I think, is the first place where mathematics teachers from all parts of the country can communicate with one another. The objective of this workshop is to develop materials which can be used in a classroom setting, such as:

- 1) materials to present ideas to students
- 2) materials to be used by students for self-teaching
- 3) materials to introduce *Mathematica* as a programming language

Thorough examination as to whether or not all of the materials developed at workshops can be used in all kinds of schools will be necessary. In Japan, it seems to have become a requirement in the mathematics curriculum to not only utilize computers in mathematics, but to also teach programming in information science classes. In this presentation, I talk about how the *Mathematica* Natsu no Gakkou program is run, some of the materials made there, the problems we confronted, and plans for developing teaching materials in the future.

south.rotol.ram.fi/~keranen/IMS99/paper41/ims99paper41.pdf

south.rotol.ramk.fi/~keranen/IMS99/paper41/ims99paper41.nb

■ Compound Program Packages in Random Science Training and Technical Modeling

I.E. Poloskov

Perm State University

Bukirev st.15

Perm, GSP, 614600, Russia

Igor.Poloskov@psu.ru

This paper presents a review of our applications of computer algebra systems (CAS) [10, 11, 40] to different problems: for analysis of random phenomena in nonlinear dynamical systems and primary education in this sphere; for finding of control functions; and for mathematical modeling of a mechanical object's movement. To archive final results for various tasks, it is required to process a large number of mathematical calculations. Moreover, the complexity of investigated objects requires automation of a procedure of model construction that results in the necessity of using CAS (together with numerical accounts) at all stages of scientific research.

south.rotol.ramk.fi/~keranen/IMS99/paper47/ims99paper47.pdf

■ “You’re Doing Simulations with Your Students, So Why Are You Using *Mathematica*?”

Phil Ramsden

Department of Mathematics

Imperial College of Science, Technology and Medicine

London, UK

At Imperial College, staff from the mathematics and chemistry departments have set up computer-based “Mathematics Laboratories” for first year chemistry undergraduates. We’ve provided several types of activity in these laboratories, but one of the most important kinds involves students setting up and studying models and simulations of chemical conditions and processes. All our simulations are implemented entirely in *Mathematica*. *Mathematica* works for us as simulation and modeling software because it allows us to be open in our approach to design. All the code that is used to set up and run the simulations is visible to the students, and indeed we require them to engage with it directly. The relationships between the underlying mathematics, the chemistry that is being modeled, and the computer code that implements the model can be explicit rather than hidden. Students can be encouraged to adopt a critical attitude toward mathematical models and to develop their own ideas and approaches. On the other hand, *Mathematica*’s power and range enable us to set up fairly realistic and uncontrived problems, and to place some of the mathematical elements of these problems firmly in the background if we choose. I illustrate these strengths of *Mathematica* using the example of a series of assessed problems. These concern the forces between ions (modeled as Newtonian point charges) and the chemical dynamics to which these forces give rise. I show examples of closed and open-ended tasks, and present excerpts of students’ work on these tasks.

south.rotol.ramk.fi/~keranen/IMS99/paper2/ramsdn.pdf

www.mathematica-journal.com/issue/v7i3/columns/ramsdn

■ Mathematical Statistics with *Mathematica*

Colin Rose

Theoretical Research Institute, Sydney, Australia

colin@tri.org.au

Murray D. Smith

University of Sydney, Australia

Murray.Smith@econ.usyd.edu.au

We present a unified approach for doing mathematical statistics with *Mathematica*. At one extreme, our PDF package empowers even the “statistically chal-

lenged” with the ability to perform complicated operations without realizing it. At the other extreme, it enables the professional statistician to tackle tricky multivariate distributions, generating functions, transformations, symbolic maximum likelihood estimation, unbiased estimation, checking (and correcting) of textbook formulae, and so on. By taking full advantage of the latest Version 4 assumptions technology, the PDF package can produce exceptionally clean and neat symbolic output.

south.rotol.ramk.fi/~keranen/IMS99/paper36/Rose.pdf
south.rotol.ramk.fi/~keranen/IMS99/paper36/Rose.nb

■ Interactive Image Processing Front End Using *MathLink*

Junzo Sato

RoboDog, Japan

Chikara Miyaji

University of Tsukuba, Japan

Mariusz Jankowski

University of Southern Maine, USA

It is not easy for commercial image-processing software to add image-processing effects or create new functions by combining tools. *Mathematica* facilitates writing extensions because mathematical expressions of image processing are achieved and functions are defined simply. On the other hand, the existing notebook front end lacks interactive processing such as direct manipulation of the graphics image with a mouse. Thus the authors have developed, using *MathLink*, an Interactive Image Processing Front End (for short, Image Processing Front End) with a variety of extensions. With this Image Processing Front End, the image processing can be done interactively and *Mathematica*'s programming ability is effectively used.

Advantages of this software are as follows:

- 1) A mouse and menus are available to draw pictures, similar to painting software.
- 2) It can read and write various graphic formats of image files.
- 3) Image-processing operations are defined as *Mathematica* functions.
- 4) Responses of a mouse and menus are defined as *Mathematica* functions.

south.rotol.ramk.fi/~keranen/IMS99/paper38/ims99paper38.pdf
south.rotol.ramk.fi/~keranen/IMS99/paper38/ims99paper38.nb

■ Recent Extensions in the Computation of Optimal Simplified Models for Systems with Time Delays

Femi Taiwo, Emem Effanga, Segun Odusanya

Chemical Engineering Department

Obafemi Awolowo University

Ile-Ife, Nigeria

ftaiwo@oauife.edu.ng

The earlier optimal method [1, 2] for the simplification of single variable systems having ordinary polynomial denominators has been extended to the multivariable case encompassing situations where the original or simplified model may have delays in their states. The performance index used for optimization is the integral of the time-weighted squared error between the responses of the original and simplified models. The performance index is first expressed in terms of the simplified model unknown parameters and a minimization of this index gives the simplified model optimal parameters. The results of this work find additional application in obtaining the closed-form expressions of the error squared integral between the reference input and output response for certain closed-loop systems involving plants with time delays.

south.rotol.ramk.fi/~keranen/IMS99/paper50/ims99paper50.pdf

south.rotol.ramk.fi/~keranen/IMS99/paper50/imas1.doc

■ Experiments in the Theory of Surfaces

Yoshihiko Tazawa

Tokyo Denki University

tazawa@cck.dendai.ac.jp

The purpose of this article is to show the advantage of using *Mathematica* in the theory of surfaces. The examples in the classical differential geometry, namely the theory of curves and surfaces, have been confined to a small group of calculable objects, because of the difficulty in evaluating geometrical quantities and solving differential equations explicitly. But *Mathematica* has made it possible to deal with a wide range of objects and to perform experimental treatments of them, based on the power of numerical calculation—the **NDSolve** and **NIntegrate** commands in particular—and graphical visualization by the **ParametricPlot** command. I would like to introduce here several animations and figures that I use in my differential geometry class for undergraduate students. First of all, these graphics help students understand the basic notions of differential geometry. Secondly, we can experiment on geometry through these graphics. This article is one of the serial talks given by the author at 1995 Developer

Conference, *IMS '97* in Rovaniemi, and *WMC 98* in Chicago. They are all targeted for the experimental usage of *Mathematica* on differential geometry. This time the topics are focused on the surfaces in the three-dimensional Euclidean space. The commands are contained in six notebooks. If the commands in other notebooks cause any trouble, please restart *Mathematica*. See also the explanation of the notebooks.

south.rotol.ramk.fi/~keranen/IMS99/paper37/ims99paper37.html

■ Teaching High School Mathematics over a Network with *Mathematica*

Kenji Yoshida

Konan Boys' High School, Japan

Shinya Ohashi

Kashiwa High School, Japan

Hiroshi Kimura

Kyushu Institute of Technology, Japan

Chikara Miyaji

Tsukuba University, Japan

Many reports studying the use of *Mathematica* in the teaching of high school mathematics have been published. The majority of these reports focuses on the benefits of using *Mathematica* in the classroom. For example, the use of *Mathematica* enables students to perform complicated calculations, make assumptions and predictions based on the results derived from these computations, and then verify their findings. This represents a rather experimental method in mathematics instruction. Before the advent of *Mathematica*, students who were not proficient in calculations spent a majority of their time performing the computations, and very little time analyzing and processing the results. However, with the aid of *Mathematica*'s computational power, students can now focus on and enjoy the thinking and analytical part of the process—something that is completely independent of their ability to calculate. In this respect, *Mathematica* is a powerful tool for studying mathematics and especially useful for those students whose strength lies not in their computational capabilities. In this article, we will introduce the networked *Mathematica* system and how, in its development, we focused on its ease of use in classroom instruction.

south.rotol.ramk.fi/~keranen/IMS99/paper42/ims99paper42.pdf
south.rotol.ramk.fi/~keranen/IMS99/paper42/ims99paper42.nb

■ Refined Constitutive Shell Equations with *MathTensor*

Ryszard A. Walentynski

*Department of Building Structures Theory
Silesia University of Technology
ul. Akademicka 5
PL44-101 Gliwice, Poland
rwal@kateko.bud.polsl.gliwice.pl*

This paper presents the results of research on refinement of constitutive shell equations with the *MathTensor* package of the *Mathematica* system. The obtained results satisfy the last equation of equilibrium and better describe the shell behavior.

south.rotol.ramk.fi/~keranen/IMS99/paper13/no13pdf.pdf

■ The Application of Winding Numbers to the Interior Problem of Nonconvex Polygons and Polyhedra

Stefan Welke

*In der Wehrbecke 29
D-53125 Bonn, Germany
Spwelke@aol.com*

The winding number for closed curves and polygons in the Euclidean plane is generalized to polyhedra in three-space. The winding number allows us to decide whether a point is in the interior or exterior of a closed polygon/polyhedron. Polygons and polyhedra need not be convex and self-intersections are allowed.

south.rotol.ramk.fi/~keranen/IMS99/paper28/Winding_Numbers_Article.nb

■ Exploring the Theory of Geometric Bifurcation

Jay H. Wolkowsky

*Department of Mathematics
University of Colorado
Boulder, Colorado 80309, USA
wolkowis@euclid.colorado.edu*

This paper will deal with the Theory of Geometric Bifurcation, which the author developed in 1986. The theory developed in those papers is very general and

abstract. So, until a symbolic software program such as *Mathematica* came along, it was very difficult to examine concrete examples that would illustrate and explain the theory. In this paper we will look at examples of bifurcating branches of solutions of the nonlinear: algebraic equations, ordinary differential equations, and partial differential equations.

south.rotol.ramk.fi/~keranen/IMS99/paper3/ims99-jhw.pdf

south.rotol.ramk.fi/~keranen/IMS99/paper3/ims99-jhw.nb

■ Transforming Systems of PDEs for Efficient Numerical Solution

Andreas Wrangsjö, Peter Fritzson, K. Sheshadri

Department of Computer and Information Science

Linköping University

S-581 83 Linköping, Sweden

andwr@ida.liu.se; petfr@ida.liu.se; sheshb@ida.liu.se

A *Mathematica* package to deal with a system of partial differential equations (PDEs) is presented. This package uses explicit finite-difference schemes to handle equations in an arbitrary number of variables that are functions of one spatial variable and time. The code has the flexibility to incorporate any difference approximation specified by the user, and transforms the given system of PDEs into a system of difference equations that can be iteratively solved using the discretized forms of initial and boundary conditions. The iteration is made considerably faster by converting the *Mathematica* code into an optimized C++ code using the MathCode C++ compiler [1]. Examples are presented in which the generated C++ code runs about a thousand times faster than the *Mathematica* code.

south.rotol.ramk.fi/~keranen/IMS99/paper7/Transformations.pdf

south.rotol.ramk.fi/~keranen/IMS99/paper7/Transformations.nb

Classifieds

Classified advertising is provided for *Mathematica* users. Rates are available at www.mathematica-journal.com/advertising.

Updated Version of Global Optimization Now Available—Global Optimization 4.0 is a collection of functions for global nonlinear optimization. It uses the *Mathematica* system as an interface for defining the nonlinear system to be solved and for computing function numeric values. Any function computable by *Mathematica* can be used as input, including degree-of-fit of a model against data, black-box functions, and simulation models. In use since early 1998, the package is well-tested by a worldwide community of users. Seven functions are provided in the package.

More info: www.wolfram.com/products/applications/globalopt

Custom Calculus Solutions Instantly—Automatic step-by-step derivatives and integrals with each step explained. Long multiplication and division of polynomials, too.

More info: www.calc101.com

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 non-commercial 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.

Do not send submissions simultaneously to *MathSource* and *The Mathematica Journal*.

Currently, *The Mathematica Journal* is operating with a backlog of articles and with a review cycle of one to one-and-a-half years.

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:

Web

Articles, graphics, and other files may be uploaded on the Internet at www.mathematica-journal.com.

Email

Articles and other material may be sent by email to editor@mathematica-journal.com. Do not email articles or attachments that exceed 1MB.

Hard Copy and/or 3.5" Diskettes 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 submitted in Macintosh or DOS format.

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 make good use of text, sections, 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:

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. footnotes in order of citation (avoid footnotes if possible);
3. references (see below);
4. a list of additional electronic material, including, if appropriate, information on the necessary environment (machine, operating system, and so on);
5. 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.

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 non-specialists 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 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.

