ECCAD 2004
East Coast Computer Algebra Day

May 8, 2004. **Wilfrid Laurier University, Waterloo, ON, Canada**
http://www.cargo.wlu.ca/eccad2004/ **eccad2004@wlu.ca**

Organizer

Ilias S. Kotsireas
Wilfrid Laurier University
Email: ikotsire@wlu.ca

Computer Algebra Research Group
of
Wilfrid Laurier University

Invited Speakers

Mathematics by Experiment: Plausible Reasoning in the 21st Century.

Prof. Jonathan M. Borwein
Dalhousie University, Canada

Implicitation and Commutative Algebra.

Prof. David A. Cox
Amherst College, USA

Solving Zero-Dimensional Systems of Equations and Inequations, Depending on Parameters.

Prof. Daniel Lazard
Université Paris 6 , France

Advisory Committe

C. W. Brown, United States Naval Academy
B. Char, Drexel University
R. M. Corless, University of Western Ontario
S. Gao, Clemson University
K. O. Geddes, University of Waterloo
M. W. Giesbrecht, University of Waterloo
E. Kaltofen, North Carolina State University
B. D. Saunders, University of Delaware
W. Sit, City University of New York
S. M. Watt, University of Western Ontario

Mathematics of Computer Algebra and Analysis (MOCAA Workshop)

May 6, 7, 2004, University of Waterloo
Waterloo, ON, Canada

Sponsor: Fields Institute

THE FIELDS INSTITUTE
FOR RESEARCH IN MATHEMATICAL SCIENCES

Organizer:
George Labahn, University of Waterloo

Organizational Committe:
Keith O. Geddes, University of Waterloo
Mark W. Giesbrecht, University of Waterloo
Arne Storjohann, University of Waterloo
Eugene Zima, University of Waterloo

Local Arrangements

D. Butcher, Computer Algebra Research Group
J. Cousineau, Wilfrid Laurier University
T. Ji, Wilfrid Laurier University
E. Lau, Computer Algebra Research Group
S. Lee, Wilfrid Laurier University
L. Schmalz, Wilfrid Laurier University
A. Zaidi, Wilfrid Laurier University

ECCAD'2004 Sponsors
- Wilfrid Laurier University Research Office
- Wilfrid Laurier University Faculty of Science
- Maplesoft Inc.
- National Science Foundation
- Natural Science and Engineering Reseach Council
- Shared Hierarchical Academic Research Computing Network

LAURIER FACULTY OF SCIENCE
Maplesoft command the brilliance
NSF
NSERC CRSNG
SHARCNET

ECCAD 2004
East Coast Computer Algebra Day

May 8th, 2004 @ Wilfrid Laurier University, Waterloo, Ontario, Canada

ECCAD, East Coast Computer Algebra Day, is an annual one-day conference that provides opportunities to learn and share new developments and to present research results in the area of Symbolic Computation. **ECCAD 2004**, will be held on **Saturday, May 8th, 2004** and is hosted by the Computer Algebra Research Group, Department of Physics and Computer Science, Wilfrid Laurier University, Waterloo, Ontario, Canada. Previous ECCAD conference homepages can be consulted at: http://www.cis.udel.edu/~saunders/eccad/.

1. **Tuesday, June 1, 2004:** e-ECCAD'2004 **An Electronic Conference in Computer Algebra**
2. **Monday, May 3, 2004: Directions to the ECCAD'2004 conference building:**
   - ECCAD'2004 will be held in the Laurier Faculty of Science Building, Room N1001, Ground Floor.
   - The Laurier Faculty of Science Building is located on King Street North, close to the King Street North and University Avenue West intersection in Waterloo. Taxi drivers (and everybody else in Waterloo) refer commonly to this intersection as "King and University".
   - The mailing address of the Laurier Faculty of Science Building is "75 University Avenue West" but the two entrances to the building (which are closer to room N1001) are actually on King Street, directly across the street from Lodge Street.
   - The Laurier Faculty of Science Building is shown as building number 24 in map1 and map2. See also this map.
3. **April 30, 2004: The conference web page will be inaccessible on Sunday May 2, 2004, due to a power shutdown affecting the Science building from 6 a.m. to 6 p.m.**
4. **April 21, 2004: NEW Deadline for submitting poster abstracts - *May 1, 2004.***
5. **April 8, 2004: All three ECCAD'2004 invited speakers are confirmed.**

The MOCAA'2004 **workshop is a two day workshop sponsored by the Fields Institute held in conjunction with ECCAD'2004. The workshop will consist of four featured invited talks along with a number of contributed talks all held in the Davis Center at the University of Waterloo.**

Maplesoft is offering two free Maple tutorials for ECCAD 2004 attendees. Two concurrent sessions are running from 4:30 to 6:00 p.m. Titles include:

1. Getting the most out of Maple's LinearAlgebra Package
   Dave Linder, Team Lead, Mathematical Software
2. Maplets - A Customizable Interface to Maple
   Stephen Forrest, Developer, Mathematical Software

Click here for more details and Maple tutorials registration information

Topics

- Algebraic Algorithms
- Symbolic-Numeric Algorithms
- Mathematical Communication
- Complexity of Algebraic Problems
- Symbolic and Numerical Linear Algebra
- Applications of Symbolic Computation

First Announcement and Call for Paper

- ECCAD'04 1st Announcement in PDF format
- ECCAD'04 1st Announcement in Text

Conference Poster

- Snapshot JPEG (800 x 1178, 200K)
- Full size JPEG (1800 x 2650, 1MB)

Important Dates

- ECCAD 04 - *Sat., May 8th, 2004*
- NEW Deadline for submitting poster abstracts - *May 1, 2004*
- Deadline for travel support - *April 30th, 2004*
- Registration begins - *January 2nd, 2004*

Travel Support

Limited funding for participants based in Canada and the United States is available, with a preference towards graduate students and new PhD's who are presenting a poster. More details will be available when the registration begins.

Information

Direct your questions to the ECCAD'2004 organizer **Ilias Kotsireas** by e-mail at ikotsire@wlu.ca or at eccad2004@wlu.ca. Phone numbers: 1-519-589-1323 and 1-519-884-0710 x 2218.

Last updated: March 11th, 2004

ECCAD 2004
East Coast Computer Algebra Day

May 8th, 2004 @ Wilfrid Laurier University, Waterloo, Ontario, Canada

CARGO

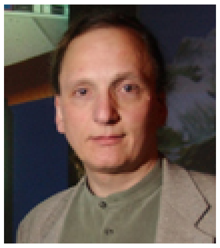
ECCAD'2004 Program

8:00 a.m. - 9:00 a.m.	ECCAD'2004 Registration
9:00 a.m. - 9:10 a.m.	Opening Remarks and Welcome Speech, Bob Rosehart, President, Wilfrid Laurier University
9:10 a.m. - 10:10 a.m.	ECCAD'2004 Invited talk, Daniel Lazard
10:10 a.m. - 10:40 a.m.	Coffee break
10:40 a.m. - 11:40 a.m.	ECCAD'2004 Invited talk, Jonathan Borwein
11:40 a.m. - 1:30 p.m.	Lunch
1:30 p.m. - 2:30 p.m.	ECCAD'2004 1st poster and demo session
2:30 p.m. - 3:30 p.m.	ECCAD'2004 Invited talk, David Cox
3:30 p.m. - 4:30 p.m.	ECCAD'2004 2nd poster and demo session
4:30 p.m. - 6:00 p.m.	Maple Tutorials
6:00 p.m	ECCAD'2004 Dinner, Dinner Speaker, Stephen Watt

ECCAD 2004
East Coast Computer Algebra Day

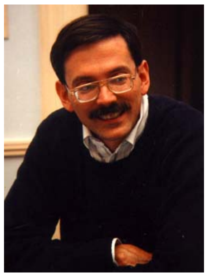
May 8th, 2004 @ Wilfrid Laurier University, Waterloo, Ontario, Canada

ECCAD'2004 Invited Speakers



Prof. Jonathan M. Borwein, FRSC — Dalhousie University, **CANADA**

Mathematics by Experiment: Plausible Reasoning in the 21st Century.



Prof. David A. Cox — Amherst College, **USA**

Implicitization and Commutative Algebra

Prof. Daniel Lazard — Université Paris 6, **FRANCE**

Solving zero-dimensional systems of equations and inequations, depending on parameters.

ECCAD 2004
East Coast Computer Algebra Day

May 8th, 2004 @ Wilfrid Laurier University, Waterloo, Ontario, Canada

- Home
- Program
- Invited Speakers
- Registration
- Accommodation
- Organization
- Sponsors

CARGO

Registration

Registration has began as of January 2nd, 2004. Interested parties have until April 20th, 2004 to register. Please consult our conference announcement for further details.

Name:	
Email:	
CC yourself:	No
Affiliation:	
Title of your poster:	
Abstract:	
Comments:	
Conference Dinner:	Yes
	Send Reset

ECCAD 2004
East Coast Computer Algebra Day

May 8th, 2004 @ Wilfrid Laurier University, Waterloo, Ontario, Canada

Accommodation

Hotels

1. University of Waterloo Student Residence rooms. Contact: Ron Eydt Village Box 16610 University of Waterloo Waterloo, Ontario, Canada N2J 4C1 Phone: 519-884-5400 Fax: 519-746-7599 e-mail: accombook@uwaterloo.ca
2. Destination Inn A block of 20 rooms has been reserved. Mention ECCAD'2004 when booking your room, to obtain the $85 conference rate. Phone: 1 (519) 884-0100. The hotel is located a $7 taxi ride from the conference site. Cut-off date for reservations: April 20, 2004. Double rooms are available. Destination Inn Information.
3. Comfort Inn Waterloo. A block of 20 rooms has been reserved for ECCAD'2004 participants. The hotel is an 1-minute walk from the conference building. Mention the **Group Number 100758** and the **Group Name ECCAD2004** when you book your room, in order to get the conference rate $88. Bookings can be made by phone at 1 (519) 747-9400 or by fax at 1 (519) 747-2134. Don't use 1-800 numbers to book your room, because you will be unable to obtain the conference rate of $88. Cut-off date for reservations: April 25, 2004. Contact Person: Diana Zalevich.
    1. Comfort Inn Information.
    2. Comfort Inn Photo.
4. Best Western Waterloo
5. The Waterloo Inn Rooms available on a first-come first-served basis. Contact Information

Transportation

Toronto Pearson International Airport

- http://www.lbpia.toronto.on.ca/

Driving Directions

To reach Laurier from:

1. Highway 401, take exit 278B (coming from London) or 278 (coming from Toronto)
2. Follow Highway 8 West to Highway 85 North (formerly 86 North) - Waterloo
3. Take the exit University Avenue West
4. On the fifth traffic light, turn left on King Street.
5. Distance from Highway 401 to campus is approximately 16 km.

- Direction Map: http://www.wlu.ca/wlu-hp/about/maps/directions.shtml
- From MSN Maps: Link
- MapQuest: http://www.mapquest.com

Bus

Airways Transit (Waterloo Division)
Address: 99A Northland Road, Waterloo, Ontario, CANADA, N2V 1Y8
Tel: (519) 886-2121
Fax: (519) 886-2141
Website: www.airwaystransit.com

Email: infowaterloo@airwaystransit.com

Additional Information

- Campus Map of WLU: http://www.wlu.ca/parking/lots.shtml
- K-W Tourism: http://www.kw-visitor.on.ca/

ECCAD 2004
East Coast Computer Algebra Day

May 8th, 2004 @ Wilfrid Laurier University, Waterloo, Ontario, Canada

Home
Program
Invited Speakers
Registration
Accommodation
Organization
Sponsors

ECCAD'2004 Organization

Organizer

Ilias S. Kotsireas
Department of Physics and Computer Science
Wilfrid Laurier University
Waterloo, Ontario, Canada
Office: 1-519-884-0710 x 2218
Website: http://sauron.wlu.ca/physcomp/ikotsireas/
E-mail: ikotsire@wlu.ca

Advisory Committee

- Christopher W. Brown, United States Naval Academy, Annapolis, MD, USA
- Bruce Char, Drexel University, Philadelphia, USA
- Robert M. Corless, University of Western Ontario, Ontario, Canada
- Shuhong Gao, Clemson University, Clemson, South Carolina, USA
- Keith O. Geddes, University of Waterloo, Waterloo, Ontario, Canada
- Mark W. Giesbrecht, University of Waterloo, Ontario, Canada
- Erich Kaltofen, North Carolina State University, Raleigh, NC, USA
- B. David Saunders, University of Delaware, Newark, DE, USA
- William Sit, City University of New York, New York, NY, USA
- Stephen M. Watt, University of Western Ontario, Ontario, Canada

Local Arrangements

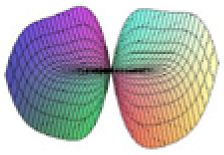
- Daniel Butcher, Computer Algebra Research Group
- Zhuliang Chen, SCG, University of Waterloo
- Jason Cousineau, Wilfrid Laurier University
- Tianying Ji, Wilfrid Laurier University
- Edmond Lau, Computer Algebra Research Group
- Spencer Lee, Wilfrid Laurier University
- Lorna Schmalz, Faculty of Science, Wilfrid Laurier University
- Azar Shakoori, ORCCA, University of Western Ontario
- Clare So, ORCCA, University of Western Ontario
- Yuzhen Xie, ORCCA, University of Western Ontario
- Asif Zaidi, Wilfrid Laurier University

ECCAD 2004
East Coast Computer Algebra Day

May 8th, 2004 @ Wilfrid Laurier University, Waterloo, Ontario, Canada

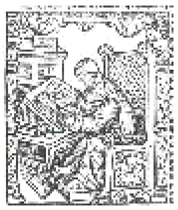
- Home
- Program
- Invited Speakers
- Registration
- Accommodation
- Organization
- Sponsors

ECCAD'2004 Sponsors



ECCAD'2004 is pleased to acknowledge generous sponsorships from:

- **Wilfrid Laurier University Research Office** (http://www.wlu.ca/research)
- **Wilfrid Laurier University Faculty of Science** (http://www.wlu.ca/science)
- **Maplesoft Inc.** (http://www.maplesoft.com/)
- **National Science Foundation, NSF** (http://www.nsf.gov)
- **Shared Hierarchical Academic Research Computing Network, SHARCNET**(http://www.sharcnet.ca/)
- **Natural Sciences and Engineering Research Council, NSERC** (http://www.nserc.ca/)
- **Ontario Research Centre for Computer Algebra** (http://www.orcca.on.ca)
- **Mathematics of Information Technology and Complex systems, MITACS** (http://www.mitacs.ca/)
- **TechnicalMastery.com Corporation** (http://www.TechnicalMastery.com)



EAST COAST COMPUTER ALGEBRA DAY 2004

ECCAD'2004

FIRST ANNOUNCEMENT AND CALL FOR POSTERS

The 11th annual ***East Coast Computer Algebra Day*** (ECCAD'2004) will be held on Saturday, **May 8, 2004**. It will be hosted at

Wilfrid Laurier University
Waterloo ON, Canada

Web Site: http://www.cargo.wlu.ca/eccad2004/

E-Mail: eccad2004@wlu.ca

Location: Amphitheater N1001, Building of the Faculty of Science,
75 University Avenue West, Waterloo, Ontario N2L 3C5 CANADA

THEMES

1. Algebraic Algorithms
2. Symbolic-Numeric Computation
3. Computer Algebra Systems and Generic Programming
4. Mathematical Communication
5. Complexity of Algebraic Problems
6. Symbolic and Numerical Linear Algebra
7. Applications of Symbolic Computation

INVITED SPEAKERS

1. Prof. Jonathan M. Borwein, FRSC, Dalhousie University, CANADA
   Title: *Mathematics by Experiment: Plausible Reasoning in the 21st Century*.

2. Prof. David A. Cox, Amherst College, USA
   Title: *Implicitization and Commutative Algebra*.

3. Prof. Daniel Lazard, Universite Paris 6, FRANCE
   Title: *Solving zero-dimensional systems of equations and inequations, depending on parameters*.

REGISTRATION

The ECCAD'2004 Registration web page can be accessed from the conference web site or through direct link: http://www.cargo.wlu.ca/eccad2004/registration.php
The registration is now open, please register as soon as you decide to attend.

POSTER SESSIONS

In keeping with tradition, there will be two poster sessions offering all participants an opportunity to present timely research in an informal environment. To submit a poster, send a title and abstract by e-mail to eccad2004@wlu.ca. If you would like to set up a demo at the conference, please specify your space and other requirements by e-mail to eccad2004@wlu.ca. All submissions are due by **April 20, 2004**.

Recent post-docs and graduate students are particularly encouraged to submit posters and arrange to set up a demo of their work.

Poster abstracts may be up to 2 pages. Poster abstracts received before the submission deadline of **April 20, 2004** will be included in the ECCAD'2004 book of abstracts which will be available at the conference.

ACCOMMODATIONS

Visit the conference web page to book your hotel. Mention ECCAD'2004-WLU when completing your reservation.

TRAVEL SUPPORT

Subject to successful funding, a limited number of participants will be supported to attend ECCAD'2004. We have requested support for US participants. Support may cover only partially your travel expenses and lodging for up to two nights. There will be no stipends. If you plan to attend and apply for support, please send your inquiry with an itemized estimate of your expenses to eccad2004@wlu.ca. Note that if your application is approved and if funding is available, you will receive your reimbursement after the meeting. All applications for support are due by **April 30, 2004**.

ORGANIZATION

Advisory Committee:

1. Christopher W. Brown, United States Naval Academy, Annapolis, MD, USA
2. Bruce Char, Drexel University, Philadelphia, USA
3. Robert M. Corless, University of Western Ontario, Ontario, Canada
4. Shuhong Gao, Clemson University, Clemson, South Carolina, USA
5. Keith O. Geddes, University of Waterloo, Waterloo, Ontario, Canada
6. Mark W. Giesbrecht, University of Waterloo, Ontario, Canada
7. Erich Kaltofen, North Carolina State University, Raleigh, NC, USA
8. B. David Saunders, University of Delaware, Newark, DE, USA
9. William Sit, City University of New York, New York, NY, USA
10. Stephen M. Watt, University of Western Ontario, Ontario, Canada

Organizer:

Ilias S. Kotsireas, Wilfrid Laurier University, Waterloo, ON, Canada

Local Arrangements Organizing Committee:

Daniel Butcher, Edmond Lau, Lorna Schmalz
Wilfrid Laurier University, Waterloo, ON, Canada

ECCAD'2004 Sponsors:

1. Wilfrid Laurier University Research Office (http://www.wlu.ca/research)
2. Wilfrid Laurier University Faculty of Science (http://www.wlu.ca/science)
3. Maplesoft Inc. (http://www.maplesoft.com/)
4. National Science Foundation, NSF (http://www.nsf.gov)
5. Natural Sciences and Engineering Research Council, NSERC (http://www.nserc.ca/)

OTHER INFORMATION

1. Up-to-date information regarding accommodations, directions, area restaurants, conference dinner, reception, registration, etc. will be posted at the conference web site:

   http://www.cargo.wlu.ca/eccad2004/

2. ECCAD'2004 will be co-located with a MOCAA Workshop.

   MATHEMATICS OF COMPUTER ALGEBRA AND ANALYSIS
   (MOCAA)

   A workshop on the mathematics of computer algebra and analysis will be held on Thursday, Friday and Saturday, May 6, 7, 8, 2004 at the University of Waterloo. Sponsored by the Fields Institute this workshop will complement the ECCAD'2004 conference held on May 8 at Wilfrid Laurier University. In particular the two events will share joint poster sessions, held at Wilfrid Laurier University on Saturday, May 8th 2004.

   Organizer:

   George Labahn, University of Waterloo, Waterloo, Ontario, Canada

   Organizational Committee:

Keith O. Geddes,	University of Waterloo, Waterloo, Canada
Mark W. Giesbrecht,	University of Waterloo, Waterloo, Canada
Arne Storjohann,	University of Waterloo, Waterloo, Canada
Eugene Zima,	University of Waterloo, Waterloo, Canada

   Invited speakers and additional details will be announced in January.

3. An additional event is planned on Wednesday afternoon, May 5th, by Maplesoft.

Introduction to Programming Maplets & The Maple 9 Linear Algebra Package

Maplesoft tutorial sessions

Maplesoft, 615 Kumpf Drive, Waterloo ON, CANADA

Introduction to Programming Maplets

Participants will be introduced to the development of custom user interfaces for Maple, using the Maplets package. Basic techniques will be practiced, by building several Maplets from scratch with the instructor.

The Maple 9 Linear Algebra Package

This tutorial provides an overview of the capabilities which Maple has to offer in the realm of linear algebra.

PLEASE HELP COMMUNICATE THIS ANNOUNCEMENT TO ALL INTERESTED PARTIES. APOLOGIES FOR MULTIPLE RECEPTIONS.

e-ECCAD 2004
East Coast Computer Algebra Day

An Electronic Conference in Computer Algebra

Welcome to the web page of e-ECCAD'2004, the web archive of the conference ECCAD'2004 held on May 8, 2004 at Wilfrid Laurier University, Waterloo, ON, Canada. Here you can browse and/or download videos with the entire talks of the invited speakers and the overheads that they used, the book of abstracts, the conference poster, the material presented during the two Maplesoft tutorials and photos.

ECCAD'2004 Invited Speakers videos and overheads

- **e-ECCAD'2004** Jonathan Borwein Invited Talk **Video** (.avi file, 676M, 1h15min) and Overheads
- **e-ECCAD'2004** David Cox Invited Talk **Video** (.avi file, 295M, 1h6min) and Overheads
- **e-ECCAD'2004** Daniel Lazard Invited Talk **Video** (.avi file, 493M, 1h12min) and Overheads file I, file II

The 3 e-ECCAD'2004 videos were produced by Waterloo Networks.

ECCAD'2004 Book of abstracts and Poster

- ECCAD'2004 Book of abstracts front/back cover, designed by Mr. Timm Vera, Public Relations Wilfrid Laurier University
- ECCAD'2004 Book of abstracts [amended and corrected, contains a total of 65 abstracts], edited by Dr. Ilias S. Kotsireas, CARGO, Wilfrid Laurier University

- ECCAD'2004 Poster, designed by Mr. Jason Cousineau, CARGO, Wilfrid Laurier University

ECCAD'2004 Sponsor messages

- Message from Dr. Bob Rosehart, President, Wilfrid Laurier University
- Message from Dr. Adele Reinhartz, Dean of Graduate Studies and Research, Wilfrid Laurier University
- Message from Dr. Arthur Szabo, Dean of Science, Wilfrid Laurier University
- Message from Dr. Laurent Bernardin, Vice President R&D, Maplesoft
- Message from Dr. Carmen Gicante, Executive Director, SHARCnet
- Speakers Sponsors, Speakers and List of Sponsors

ECCAD'2004 Maplesoft Tutorials

- Linear Algebra New Maple 9.5 functionalities
- Maplets New Maple 9.5 functionalities

ECCAD'2004 Maple/MATLAB On-line Video-based Courses (Demo)

- Maple Mastery I
- MATLAB Mastery I

ECCAD'2004 Photos

Special Thanks: (in chronological order)

1. Edmond Lau,
2. Jürgen Gerhard,
3. Jason Cousineau,
4. Dan Butcher,
5. Lorna Schmalz,
6. Manfred Gartner,
7. Jeff Picklyk,
8. Azar Shakoori,
9. Clare So,
10. Mark Giesbrecht,
11. George Labahn,

12. Eugene Zima,
13. Rick Henderson

ECCAD'2004, EAST COAST COMPUTER ALGEBRA DAY 2004
May 8, 2004
Wilfrid Laurier University, Waterloo ON, Canada
`http://www.cargo.wlu.ca/eccad2004/` `eccad2004@wlu.ca`

ECCAD'2004 Invited Speakers:

- Prof. Jonathan M. Borwein, FRSC Dalhousie University, CANADA. Title: Mathematics by Experiment: Plausible Reasoning in the 21st Century.
- Prof. David A. Cox Amherst College, USA. Title: Implicitization and Commutative Algebra.
- Prof. Daniel Lazard Université Paris 6, FRANCE. Title: Solving zero-dimensional systems of equations and inequations, depending on parameters.

ECCAD'2004 Organizer: Ilias S. Kotsireas, Wilfrid Laurier University, Waterloo, ON, Canada
ECCAD'2004 Local Arrangements: Daniel Butcher, Jason Cousineau, Edmond Lau.
ECCAD'2004 Advisory Committee:

- Christopher W. Brown, United States Naval Academy, Annapolis, MD, USA
- Bruce Char, Drexel University, Philadelphia, USA
- Robert M. Corless, University of Western Ontario, Ontario, Canada
- Shuhong Gao, Clemson University, Clemson, South Carolina, USA
- Keith O. Geddes, University of Waterloo, Waterloo, Ontario, Canada
- Mark W. Giesbrecht, University of Waterloo, Ontario, Canada
- Erich Kaltofen, North Carolina State University, Raleigh, NC, USA
- B. David Saunders, University of Delaware, Newark, DE, USA
- William Sit, City University of New York, New York, NY, USA
- Stephen M. Watt, University of Western Ontario, Ontario, Canada

Special thanks: L. Bernardin, Maplesoft, H. Chatterton, Maplesoft, J.Dell, Maplesoft, A. Epstein, SIAM, T. Frost, Wilfrid Laurier University, M. Gaber, Wilfrid Laurier University, M. Gartner, Wilfrid Laurier University, J. Gerhard, Maplesoft, G. Gordon, Wiley, L. Hanna, Wilfrid Laurier University, B. Harrison, Wilfrid Laurier University, S. Ilie, University of Western Ontario, T. Ji, Wilfrid Laurier University, K. Manzi, Wilfrid Laurier University, J. Picklyk, Waterloo Networks, B. Prentice, Thomson Nelson, L. Schmalz, Wilfrid Laurier University, E. Smirnova, University of Western Ontario, D. Vaughan, Wilfrid Laurier University, T. Vera, Wilfrid Laurier University, L. Wei, Wilfrid Laurier University, L. Zajac, SHARCnet. *THANK YOU, MERCI BEAUCOUP*

MOCAA'2004, Mathematics of Computer Algebra and Analysis 2004
May 6, 7, 8, 2004
Wilfrid Laurier University, Waterloo ON, Canada
`http://www.scg.uwaterloo.ca/~mocaa/` `mocaa@scg.math.uwaterloo.ca`

MOCAA'2004 Invited Speakers:

- Edgardo Cheb-Terrab, Maplesoft Inc, Canada
- Robert Corless, ORCCA, University of Western Ontario, Canada
- John May, North Carolina State University, USA
- Michael Monagan, CECM, Simon Fraser University, Canada
- Gilles Villard, CNRS, ENS-Lyon, France

MOCAA'2004 Organizing Committee General Chair: George Labahn, Director, Symbolic Computation Group, School of Computer Science, University of Waterloo.
MOCAA'2004 Organizing Committee: Keith Geddes, Mark Giesbrecht, Arne Storjohann, Eugene Zima, Symbolic Computation Group, School of Computer Science, University of Waterloo.

The MOCAA 2004 workshop is a two day workshop sponsored by the Fields Institute and Mathematics of Information Technology and Complex Systems (MITACS) held in conjunction with East Coast Computer Algebra Day (ECCAD). The workshop will consist of five featured invited talks along with a number of contributed talks all held in the Davis Center at the University of Waterloo. In addition, there will be a poster session on May 8 run in conjunction with ECCAD held at Wilfrid Laurier University. The talks themselves are expected to cover a wide variety of topics of interest to researchers in computer algebra.

Message from Dr. Adele Reinhartz, Dean of Graduate Studies and Research

The Faculty of Graduate Studies and Research welcomes ECCAD'2004 to Wilfrid Laurier University, and extends hearty congratulations to the Computer Algebra Research Group in Laurier's Department of Physics and Computer Science and the conference organizers. We are extremely proud of the important and high quality research being done at Laurier in this field, and we are confident that the day will be stimulating and productive, both in terms of the research developments presented, and in the interactions among scholars.

Best wishes for a successful and enjoyable conference

Adele Reinhartz, PhD
Dean of Graduate Studies and Research
Wilfrid Laurier University

Message from Dr. Arthur G. Szabo, Dean of Science

It is my pleasure to welcome you to the 2004 East Coast Computer Algebra Day on behalf of the Laurier's Faculty of Science. It is an honour for Laurier Science to be one of the sponsors for this important event.

Our Faculty consists of six departments i.e. Biology, Chemistry, Kinesiology & Physical Education, Mathematics, Physics & Computer Science, and Psychology. We are a young Faculty (formed in 2000 from a division of the Laurier's longstanding Faculty of Arts and Science), with a strong focus on research in the natural, physical, and social sciences. While you're here, do take note of our just-opened Laurier Science Research Centre, attached to the Science Building. The Centre will be the home of research enterprise in each of our disciplines, including Computer Science.

I congratulate the organizers of the ECCAD'2004 for the high quality of the program. Welcome to the Laurier campus, and enjoy the meeting.

Arthur G. Szabo, PhD, FCIC
Dean of Science
Wilfrid Laurier University

ECCAD'2004 Maplesoft Tutorials `http://www.maplesoft.com/`

- Title: Getting the most out of Maple's LinearAlgebra Package Maplesoft Presenter: Dave Linder, Team Lead, Mathematical Software

  The LinearAlgebra package in Maple is an environment for doing symbolic and numeric linear algebra computations. Several techniques can be used to get the most out of this package, in terms of efficiency, speed, memory consumption, and results. Without requiring prior knowledge of the package, this tutorial will present a number of such techniques, with examples from such areas as linear system solving at high precision, modular linear algebra, and use of datatypes. The session will involve the participants in using Maple themselves on supplied worksheet material. The goal of the material is to allow greater understanding through discovery, both of some potential difficulties and the means to avoid them. Prior knowledge of basic mathematical linear algebra will be useful to the participant. Links and references to further reading and documents will be provided.

- Title: Maplets - A Customizable Interface to Maple Maplesoft Presenter: Stephen Forrest, Developer, Mathematical Software

  This tutorial will introduce Maplets, a customizable interface to Maple that allows an author of Maple code to create windows, dialogs, and other visual interfaces that interact with a user to provide the power of Maple in a friendlier environment. We will see how to create simple Maplets with only a few lines of code, how to add simple buttons, boxes, and menus to your Maplet, and how to build programming logic into your Maplet so that complex Maple calculations can be performed with the click of a mouse.

  After a short introduction, the session will be interactive. Attendees will have the opportunity to write their own Maplets. We will begin with a simple example and build up to the point where we can create more advanced Maplets.

Mathematics by Experiment: Plausible Reasoning in the 21^{st} Century

ECCAD' 2004 Invited Talk, Jonathan Michael Borwein, FRSC, Dalhousie University, CANADA

Abstract. *I shall talk generally about experimental and heuristic mathematics and give accessible, primarily symbolic, examples.*

The emergence of powerful mathematical computing environments like *Maple* and *Matlab*, the growing availability of correspondingly powerful (*multi-processor*) computers and the pervasive presence of the internet allow for research mathematicians, students and teachers, to proceed heuristically and 'quasi-inductively'. We may increasingly use symbolic and numeric computation visualization tools, simulation and data mining.

Many of the benefits of computation are accessible through low-end 'electronic blackboard' versions of experimental mathematics. This permits livelier classes, more realistic examples, and more collaborative learning. Moreover, the distinction between *computing* (HPC) and *communicating* (HPN) is increasingly blurred.

The unique features of our discipline make this both more problematic and more challenging. For example, there is still no truly satisfactory way of displaying mathematical notation on the web; and we care more about the reliability of our literature than does any other science. The traditional role of proof in mathematics is arguably under siege.

Illuminated by examples, I intend to pose questions—discussed at length in [4]—such as:

- What constitutes *secure mathematical knowledge*?
- When is computation convincing? Are humans less fallible?
- What tools are available? What methodologies?
- What about the 'law of the small numbers'?
- How is mathematics actually done? How should it be?
- Who cares for certainty? What is the role of proof?

And I shall offer some personal responses and assessments.

References

1. Jonathan M. Borwein and Robert Corless, "Emerging Tools for Experimental Mathematics," *American Mathematical Monthly*, **106** (1999), 889–909. [1]
2. D.H. Bailey and J.M. Borwein, "Experimental Mathematics: Recent Developments and Future Outlook," pp, 51-66 Vol. I of *Mathematics Unlimited* 2001 *and Beyond,* B. Engquist and W. Schmid (Eds.), Springer-Verlag, 2000.
3. J. Dongarra, F. Sullivan, "The top 10 algorithms," *Computing in Science & Engineering,* **2** (2000), 22–23.
4. J.M. Borwein and D.H. Bailey, *Mathematics by Experiment: Plausible Reasoning in the 21st Century,* and *Experimentation in Mathematics: Computational Paths to Discovery,* (with R. Girgensohn), AK Peters Ltd, November 2003. The website with resources for, and an extended sample of, these two books is `http://www.expmath.info`.

Implicitization and Commutative Algebra

ECCAD' 2004 Invited Talk, David A. Cox, Amherst Colege, USA

This talk will discuss implicitization using resultants, moving surfaces, and relations between the two. I will illustrate how concrete questions in implicitization leads naturally to concepts in commutative algebra such as syzygies, free resolutions, regularity, and local complete intersections. The talk will be down-to-earth but the moral would be that serious commutative algebra is involved.

Solving zero-dimensional systems of equations and inequations, depending on parameters

ECCAD' 2004 Invited Talk, Daniel Lazard, Université Paris 6, FRANCE

Let us consider a system of polynomial equations and inequations and split the set of its variables in two subsets, the set of unknowns and the set of indeterminates (this partition may be given as input but may also be done as a first step of computation). The addressed problem consists in computing the number of real solutions of the systems as a function of the parameters, under the hypothesis that, for almost all value of the parameters, the equations have only a finite number of common complex solutions.

We present a general algorithm for this problem. It has been used for solving several problems of big size, which are far outside the range of applicability of any other existing method. A noteworthy feature of this algorithm is that Gröbner bases, triangular sets and CAD (in the space of the parameters only) are all needed as subalgorithms.

[1] All references are available at `http://www.cecm.sfu.ca/preprints/`.

ECCAD'2004
Poster Abstracts
(in alphabetical order)

1 Poster Title:

From ELIMINO to OpenElimino and Beyond

Presenter:

Iyad A. Ajwa
Ashland University

ELIMINO is a new computer-mathematics research system that has been developed at the Chinese Academy of Sciences in Beijing. The focus of ELIMINO is on the implementation of Wu's Method, an advanced mathematical computation with a wide variety of applications including mechanical geometry theorem proving, theoretical physics, CAGD, robotics, and automated reasoning. Although the main focus of ELIMINO is on Wu's Method and other related algorithms, the system has the potential to become a premier computer algebra system for polynomial manipulations. The system provides general computational capabilities for numbers, polynomials, and other mathematical objects enabling researchers to perform sophisticated mathematical computations such as Characteristic Sets and Grobner Bases. ELIMINO has a clear and simple layered architecture suitable as a base for a community-based open development system. From a software engineering viewpoint, the power and applicability of ELIMINO can be significantly increased by:

1. Applying software engineering techniques such as Object-Oriented Design/Programming (OOD/OOP), design patterns, and polymorphism;
2. Re-organizing the system into Java-style packages;
3. Making it an open-source community development software;
4. Providing good documentation.

2 Poster Title:

Some Aspects of Polynomial Algebra by Values

Presenter:

Amir Amiraslani
University of Western Ontario

This poster outlines a new algorithm for finding the roots of multi-variate polynomials using the division algorithm. The point of view taken in the talk is that all polynomials considered are given by their values at known points and that the degrees of the polynomials are known or can be deduced. We choose to perform all operations directly on the given values in order to avoid potential numerical instability in changing bases; of course this prevents us from discovering and using any possible sparse representation, which might be cheaper, but our goal is to provide stable algorithms.

This is a joint poster with Dr. Rob Corless, Dr. Laureano Gonzalez-Vega and Azar Shakoori.

3 Poster Title:

Arbitrary order Symbolic Derivatives and Integrals

Presenter:

Mhenni Benghorbal
University of Western Ontario

We discuss methods for finding the n-th derivative of certain classes of functions. We recall the rational function approach of full partial fraction decomposition, and extend this by the Mittag-Leffler theorem to meromorphic functions whose residues can be calculated symbolically. We also discuss finding the nth derivative of power-exponential functions by explicit construction and solution of recurrence equations for the derivatives. These computations are useful for finding formal power series, for finding the solutions to certain fractional differential equations, for symbolic integration, and also lead to some new functional identities.

This is a joint work with Dr. Robert Corless.

4 Poster Title:

Symbolic Tools - extending GAUSS with Maple

Presenter:

Jon Breslaw
econotron.com

The concept behind Symbolic Tools is to augment the numeric and graphical capabilities of GAUSS with additional types of mathematical functionality based on symbolic computation, usig the Maple kernel. Additional functionality includes symbolic algebra, linear algebra, automatic differentiation, language extension and precision. Details and examples are available at:

http://www.econotron.com.

5 Poster Title:

Debugging a High Level Language via a Unified Interpreter and Compiler Runtime Environment

Presenter:

Jinlong Cai
University of Western Ontario

Aldor is a programming language that provides higher-order facilities for symbolic mathematical computation. Aldor has an optimizing compiler and an interpreter. The interpreter is slow, but provides a useful debugging environment. Compiled Aldor code is efficient, but cannot be debugged using user-level concepts. By unifying the runtime environments of the Aldor interpreter and compiled Aldor executables, we have realized a debugger for Aldor. This integration of the various existing functionalities in its debugger improves the development environment of Aldor in a significant manner, and provides the first such environment for symbolic mathematical computation.

We propose that this approach can be useful for other very high level programming languages.

Joint work with Marc Moreno Maza, Stephen Watt, Martin Dunstan.

6 Poster Title:

Structured Maplets

Presenter:

Colin Campbell
TechnicalMastery.com

Structured Maplets divide the Maplet code into three distinct procedures: (1) the core computation, (2) the user interface, and (3) the "glue" between these two. The result is highly readable and extensible Maplets. For example, see: `www.mapleapps.com/categories/education/physics/worksheets/RLCMaplet.mws`

7 Poster Title:

Learning Maple 24/7 with "Maple Mastery I" on-line videos

Presenter:

Colin Campbell
TechnicalMastery.com Corporation

"Maple Mastery I" teaches you Maple using on-line video-based lessons. With no other instruction, "Maple Mastery I" will empower you to solve mathematical problems on your own productively and confidently using Maple. A sample lesson may be found at:

http://www.TechnicalMastery.com.

8 Poster Title:

Pattern Mining and Code Transformation in Maple

Presenter:

Jacques Carette
McMaster University

Poster abstract We show how patterns of code, both good and bad, can automatically be mined from a large corpus of Maple code, in this case Maple's own library. We can then use code transformation techniques to modify code automatically - away from bad patterns and hopefully improving it.

9 Poster Title:

A fast implementation of rational system solving for integer matrices

Presenter:

Zhuliang Chen
University of Waterloo

We present a fast implementation of the well-known p-adic lifting algorithm for computing the exact solution vector to a nonsingular system of linear equations with integer coefficients. A feature of this problem is that the solution vector typically contains large, multi-precision rational numbers, even thousands of bits long. By combining a careful choice of the lifting modulus with a residue number system, we are able to perform the lions share of computation in 53-bit precision. This allows computing all matrix-matrix and matrix-vector products using the portable and highly optimized numerical BLAS library. On a modern workstation the algorithm will compute the exact solution to a system of dimension two thousand with single digit entries in under two minutes. The numerators and denominators of entries in the solution vector have over four thousand decimal digits.

10 Poster Title:

On Fraction-free Polynomial Matrix Computations and Matrix Multiplication

Presenter:

Howard Cheng
University of Lethbridge

At ISSAC'2003, Giorgi, Jeannerod, and Villard showed that computing a column-reduced form of a polynomial matrix over $K[x]$ can be reduced to polynomial matrix multiplication, where K is a field. The so-called σ-bases of Beckermann and Labahn were used as a tool to perform the computation, and their recursive properties were exploited. Potential coefficient growth is not examined. In this poster, we study similar computations over $Q_D[x]$, where Q_D is the quotient field of an integral domain D. Order bases (special cases of σ-bases) and the corresponding Fast Fraction-Free Gaussian (FFFG) elimination algorithm are used in controlling coefficient growth in these computations. However, order bases do not possess the same recursive properties as σ-bases. Using modified Schur complements, we show how polynomial matrix multiplication can be incorporated into the FFFG algorithm in some cases.

This work was done jointly with George Labahn (University of Waterloo).

11 Poster Title:

When a matrix is a resultant matrix

Presenter:

Arthur D. Chtcherba
University Of Texas - Pan American

In recent years a number of methods for computing the resultant of a polynomial system have been proposed in the literature. In most cases, the resultant is extracted from the determinant of a non-singular symbolic matrix. Some of the methods relax the non-singularity requirement on the matrix, and instead use the determinant of the biggest non-singular submatrix (maximal minor). Such a relaxation often uses specific properties of the matrix construction used in specific methods. This paper is an attempt to establish conditions on a given symbolic matrix (independent of any construction method) so that the resultant of the polynomial system can be extracted from the determinant of its maximal minor.

For some constructions, these conditions are easy to verify, for example, in case of the RSC construction based on the generalized Cayley-Dixon formulation as proposed by Kapur, Saxena and Yang (1994), the Dixon matrix has to have an independent column. But in general, for a given symbolic matrix to be a resultant matrix, one needs to check if the solutions of the linear system defined by the symbolic matrix include the solutions of the polynomial system for the specialization making the resultant vanish.

The insights developed in this paper offer a number of useful applications. Given a resultant matrix, these conditions can be used to further reduce the matrix size and hence, improve the efficiency of the symbolic determinant computation; it is also possible to develop methods where a resultant matrix is constructed incrementally using the Newton polytopes of the polynomials in a polynomial system, similar to Emiris and Canny, (1995), or using the support hull of the polynomials of the polynomial system as proposed in Chtcherba and Kapur, (2004). Conditions developed in the paper are also likely to facilitate the development of new resultant methods since the conditions on a symbolic matrix to be a resultant matrix are much weaker, compared to conditions derived from specific constructions. The proposed conditions also provide uniform generic way to consider dialytic, non-dialytic as well hybrid matrices which can serve as resultant matrices.

12 Poster Title:

The numerical evaluation of the Wright w function

Presenter:

Hui Ding
University of Western Ontario

This poster describes the implementation of the Wright w function. It shows some main properties of w, and details the numerical evaluation of w over C.

13 Poster Title:

Optimizations for deeply nested parametric types

Presenters:

Laurentiu Dragan
Stephen Watt
Ontario Research Centre for Computer Algebra (ORCCA) University of Western Ontario

Aldor programming language was initially developed as a programming environment for AXIOM computer algebra system. Aldor relies heavily on deeply nested generic types to represent algebraic domains. The usage of generic types is very important from the code re-usability viewpoint, but unfortunately the code reusability comes at the cost of efficiency. Computer algebra systems require that operations based on the types that are constructed with deeply nested types to be very efficient. The goal of this work is to optimize the generic types. One of the most common form of optimization is through specialization. As such, we can specialize the algebraic types into highly specialized types. Algebraic types, represented as parametric domains in Aldor, can be specialized in only one fully or partially specialized domain on which many compiler optimizations can be applied because of extra information available due to specialization. The specialization is realized by cloning the generic type and its associated operations and by using a common optimization method, namely in-line expansion. The code from the parameters of the original type are expanded in-line into the specialized type. This optimization should also help other local optimizations that were not possible before, because most optimizations are intra-procedural. The main disadvantage of this method is the code explosion that may result after in-line expansion of the domains. A trade-off must be found between code expansion and execution speed increase. Too much code specialization may do more harm than good. Some experimental trials show very promising results, yielding improvements up to 2-3 times over the original code.

14 Poster Title:

Creating a Platform Independent Digital Ink Architecture

Presenter:

Kevin Durdle
Ontario Research Centre for Computer Algebra (ORCCA) University of Western Ontario

Since the introduction of mobile devices capable of collecting ink, handwriting recognition has evolved significantly i.e. from shorthand notations to entire language sets with accuracy rates beyond 95% on contemporary platforms.

Natural language recognizers have been able to elude new problems that arise with the introduction of mathematical ink recognition. For example, mathematics frequently requires two-dimensional space to denote complex equations while using exponentially larger symbol sets than existing languages.

While required processing power to recognize mathematics will eventually become available to mobile devices, there will be a constant demand for the ability to collaborate ink between platforms. Existing solutions permit limited viewing of ink representations between applications or platforms. Even when this is possible, there will be inadequate support for the manipulation, processing or recognition of ink beyond the platform originally collected from.

Our research focuses on the introduction of independent digital ink architecture. Such solutions would allow immediate and long term benefits to both consumers and application developers, including:

1. The possibility of real-time, intelligent collaboration between devices operating on different platforms;

2. Resource limited devices could transfer ink to servers for processing, manipulating or recognizing, returning the results to the original device;

3. Applications oriented around ink could be easily adapted to allow for the targeting of multiple paradigms.

Building on the work of previous research that has occurred at the Ontario Research Centre for Computer Algebra (ORCCA), my studies focus on the creation of a such a platform. Thanks to research provided by prior graduate students, the necessary data structures as well as the required functions have been identified. These findings will ensure that the architecture will provide platform independence while at the same time, taking advantage of existing infrastructures.

15 Poster Title:

A Grobner Walk Implementation in Maple

Presenter:

Jeff Farr
Simon Fraser University

Nearly a decade ago, Collart, Kalkbrener and Mall introduced a new method for Grobner basis conversion, the Grobner Walk. Like the well-known FGLM algorithm, the Grobner walk is used to change a Grobner basis of an ideal with a given order to a Grobner basis of the ideal with respect to a new order. Unlike FGLM, though, the Grobner Walk does not require the ideal to be zero-dimensional.

While the main ideas of the Grobner Walk are straightforward, the actual implementation of the algorithm is less understood. For example, it is not clear exactly what path should be used for ẅalkingg̈in the general case; also, it is possible that a special selection strategy for Buchberger's algorithm may further streamline the walk. As a result, the Grobner Walk has been included in only a few computer algebra packages.

Addressing these questions is important as many applications require a Grobner basis under lex order, which is much too hard to compute directly. Further, some computational data already has suggested that the Grobner Walk could outperform FGLM. We present some of these issues as they have been encountered in our preliminary implementation in Maple.

16 Poster Title:

Calculating Anomalies in Non-Commutative Field Theories Using Maple

Presenter:

Marie-Paule Gagne-Portelance
University of Western Ontario

Recently, there has been growing interest in non-commutative field theories involving the Moyal product. We wish to examine how the introduction of the Moyal product into conventional quantum field theory alters the anomalies present. The resulting calculations are laborious, involving taking the traces of products of gamma matrices and evaluating Feynman integrals containing the Moyal Product. In this poster paper, we show how an updated version of the Maple gamma matrix package HIP, as well as maple programs designed to manipulate Feynman integrals were used to perform these calculations. We conclude by comparing our result to that obtained in conventional quantum field theory.

17 Poster Title:

On Smale's 6th Problem: A solution in the four-body case

Presenter:

Marshall Hampton
University of Minnesota

In 1998 Smale formulated 18 problems for the 21^{th} century, the sixth of which was: prove that there are finitely many relative equilibria for positive masses in the planar n-body problem. In joint work with Richard Moeckel, we have proven this result for $n = 4$.

18 Poster Title:

A New Bi-orthogonalising Block Lanczos Algorithm

Presenter:

Bradford Hovinen
University of Waterloo

Blocked iterative linear system solvers are important for solving very large sparse linear systems since they can be easily parallelized. A block Lanczos algorithm that succeeds in solving a consistent linear system over large fields is a straightforward generalization of the scalar algorithm and has recently been analyzed in a paper by Wayne Eberly, to appear in ISSAC 2004. Here we propose an algorithm that is reliable over arbitrary fields, addressing the key applications of solving discrete logarithms and factoring integers using index calculus techniques.

19 Poster Title:

Error Backward for DAE

Presenter:

Silvana Ilie
University of Western Ontario

We are interested in extending the Gröbner-Alekseev nonlinear variation of constants formula to the DAE case. As in the IVP for ODE case, this would allow connection of backward error (residual error), which is computable, with forward error, which is desired. This poster reviews the proof of the Gröbner-Alekseev nonlinear variation of constants formula in the IVP case, and shows examples of why such a formula would be useful in the DAE case.

Joint work with Prof. Rob Corless and Prof. Greg Reid.

20 Poster Title:

Hadamard Ideals and Hadamard Matrices

Presenters:

Ilias S. Kotsireas
Dan Butcher
Wilfrid Laurier University

We present an overview of using high-performance computing (HPC) techniques in the study of Hadamard matrices. The theoretical basis for these computations is provided by the concept of Hadamard ideals, which is intimately related with the 1- and 2-dimensional elementary symmetric functions. The computations have been performed remotely at the Shared Hierarchical Academic Research Computing Network (SHARCNET) clusters in Ontario, Canada, a WestGrid cluster in British Columbia, Canada and at the Centre de calcul formel MEDICIS, École Polytechnique, Paris, France.

21 Poster Title:

Hadamard Matrices and Genetic Algorithms

Presenters:

Ilias S. Kotsireas
Jason Cousineau
Wilfrid Laurier University

We apply SGA (Simple Genetic Algorithm) to the study of Hadamard matrices. Hadamard matrices of order more than 50 are constructed routinely, using an objective function coming from the associated Hadamard ideals. The objective function is minimized with the aim to make it equal to zero. Genetic algorithms are a powerful and efficient algorithmic tool with a very wide range of applicability.

22 Poster Title:

Eigenvalue method for Implicitization

Presenters:

Ilias S. Kotsireas
Edmond Lau
Wilfrid Laurier University

We present some recent advances in and efficient implementations of, the eigenvalue method for implicitization. These See [2] for a recent overview of methods for exact implicitization of (algebraic) curves and surfaces. These implementations are to be incorporated into the IPSOS algorithm, see [1].

References:

[1] Ioannis Z. Emiris, Ilias S. Kotsireas. *Implicit Polynomial Support Optimized for Sparseness.* V. Kumar et al. (Eds.) ICCSA'2003 Proceedings, Montreal, Canada, LNCS 2669, pp. 397-406.

[2] Ilias S. Kotsireas. *Panorama of methods for exact implicitization of algebraic curves and surfaces.* **Geometric Computation**, World Scientific Press, 2004, F. Chen, D. Wang eds pp. 126-155.

23 Poster Title:

Implementing GIDL Bindings for Aldor

Presenter:

Michael Lloyd
Ontario Research Centre for Computer Algebra (ORCCA) University of Western Ontario

Parametric polymorphism is now becoming commonplace in main-stream programming languages. For example, both C++ and the new version of Java provide templates.

The GIDL compiler, developed in our laboratory allows software components written in different languages to work together and and take advantage of parametric polymorphism. We have developed GIDL bindings for Aldor, allowing Aldoŕs computer algebra libraries to be used from C++ and generic Java.

This poster describes the GIDL bindings for Aldor and provides details of their implementation.

24 Poster Title:

A Generalization of Gao's Factorization Algorithm to Polynomials in Many Variables

Presenter:

John May
North Carolina State University

We present a generalization of Ruppert's PDE based polynomial irreducibility test to polynomials with more than 2 variables. In the same way Ruppert's two variable condition can be turned into a factorization algorithm we turn the generalization into a factorization algorithm as well. We also discuss issues with non-squarefree and non-content free polynomials.

Our algorithm seems a little too slow to be pratical as an exact factorizer, but it shows promise to compute approximate factorizations when the input polynomials have a relatively large radius of irreducibility. We demonstate examples and a MAPLE implementation.

This is joint work with Erich Kaltofen, Zhengfeng Yang, and Lihong Zhi.

25 Poster Title:

On Polynomial GCDs over Direct Products of Fields

Presenter:

Marc Moreno Maza
University of Western Ontario

Let K be a field of multivariate rational functions over an infinite field. Let L be a direct product of fields extending K by a tower of simple algebraic extensions. We present a modular algorithm for computing polynomial gcds over L based on a Hensel lifting strategy.

The rational reconstruction is avoided by "guessing" the denominator of the gcd before the lifting step. The algorithm is probabilistic but succeeds with probability one. The cost is essentially that of the lifting step and our preliminary implementation shows a significant improvement with respect to other methods.

This is a joint work with Francois Boulier (University of Lille, France) and Cosmin Oancea (University of Western Ontario).

26 Poster Title:

Differential resolvents are complete but not rotation and translation invariant

Presenter:

John Michael Nahay
Swan Orchestral Systems

Is a differential resolvent of a polynomial a differential resolvent of the minimal polynomial of every solution of the resolvent? The answer is yes. The author recently proved this result in February 2004 for this poster. Therefore in this sense differential resolvents are complete. This proof has not been presented in the author's earlier publised research on differential resolvents in the Journal of Differential Equations (Volume 191 Issue 2 pages 323-347, 1 July 2003) or the International Journal of Mathematics and Mathematical Sciences (Volume 32 Issue 12 pages 721-738, 22 December 2002 and Volume 2004 Issue 7 pages 365-372, 1 February 2004). We use the computer algebra system Mathematica to investigate how differential resolvents of the quadratic equations of conic sections vary under rotation and translation. In particular, we determine the values of the constants for which linear combinations over constants of the solutions of resolvents form a continuous bounded curve without cusps such as an ellipse.

27 Poster Title:

An approach to using ALDOR libraries with Maple

Presenters:

Cosmin Oancea
Stephen Watt
Ontario Research Centre for Computer Algebra (ORCCA) University of Western Ontario

One of the positions held over the past two decades of mainstream computer algebra system design has been that there should be one over-arching language that serves both the end user and library developer. This has led to systems either that use modified scripting languages for their libraries (e.g. Maple), or that use modified library-building languages for their user interface (e.g. Axiom). A variant of this approach is to build much of the the mathematical support in a lower-level system implementation language, such as Lisp (e.g. in Macsyma) or C (e.g. in Mathematica).

This poster examines what is required to use Aldor libraries to extend Maple in an effective and natural way. This represents a non-traditional approach to structuring computer algebra software: using an efficient, compiled language, designed for writing large complex mathematical libraries (Aldor [7]), together with a top-level system based on user-interface priorities and ease of scripting (Maple [3]). We assume that the functionality of the extension library may either be a collection of very fast core routines or calls to larger software components. We therefore look beyond the solutions offered by loosely coupled computer algebra systems, e.g. OpenMath[4] or the software bus[5].

Our solution consists of two parts: the first part allows the low-level run time systems of Maple and Aldor to work together. That is, it allows Aldor functions to call Maple functions and vice versa, and provides a protocol whereby the garbage collectors of the two systems can interact with each other to cooperate in collecting garbage when structures span two system heaps. This low-level work has been reported elsewhere [6].

The second part, reported here, implements a high-level correspondence between Maple and Aldor concepts. The aim has been to allow Aldor domains to appear to the user as Maple modules, and to bridge the semantic differences between the two environments. For this we use a tool to generate Aldor, C and Maple code (to wrap each Aldor library) as well as supporting runtime code (to do type dispatch and caching). The resulting package, which we call MAPAL, allows standard Aldor libraries to be used in a standard Maple environment [3]. The Aldor functions run tightly coupled to the Maple environment, able to directly and efficiently manipulate Maple data objects. From user's point of view, the information of how to use the Aldor components is provided in a Maple-like fashion, while the internal invocation mechanism is completely transparent.

We see the following as contributions of the approach we outline:

- Aldor has been found to offer efficiencies comparable to hand-coded C++ [2]. This approach allows user extensions to operate with efficiencies comparable to Maple kernel routines.
- These extensions are in a high-level language, well-adapted for mathematical software, allowing the programmer to ignore lower-level details and well-integrated in the Maple environment. This is very

different from earlier work on foreign function interfaces.

- Aldor is designed for mathematical "programming in the large" and provides linguistic support for such concepts as generic algorithms, algebraic interface specification and enforcement, etc.

- Authors of Aldor code often wish to make their functionality available through Maple, e.g. Bronstein's library for algorithms on differential operators and Moreno Maza's library for triangular sets.

References:

1. L. Bernardin, B. Char, and E. Kaltofen. Symbolic computation in java: An appraisement. In Proc. ISSAC 1999, pages 237 244. ACM, 1999.

2. M. B. Monagan, K. O. Geddes, K. M. Heal, G. Labahn, S. M. Vorkoetter, J. McCarron, and P. DeMarco. Maple 9 Advanced Programming Guide. Maplesoft, 2003.

3. Special issue on OpenMath. ACM SIGSAM Bulletin, 34(2), June 2000.

4. J. Purtilo. Applications of a software interconnection system in mathematical problem solving environments. In Symposium on Symbolic and Algebraic Manipulation (SYMSAC 86), pages 16 23. ACM, 1986.

5. S. M. Watt. A study in the integration of computer algebra systems: Memory management in a Maple-Aldor environment. In Proc. International Congress of Mathematical Software, pages 405 411, 2002.

6. S. M. Watt. Aldor. In J. Grabmeier, E. Kaltofen, and V. Weispfenning, editors, Handbook of Computer Algebra, pages 154 160, 2003.

7. S. M. Watt, P. A. Broadbery, S. S. Dooley, P. Iglio, S. C. Morrison, J. M. Steinbach, and R. S. Sutor. AXIOM Library Compiler User Guide. Numerical Algorithms Group (ISBN 1-85206-106-5), 1994.

28 Poster Title:

Compiler Enforced Memory Semantics in Legacy Code via Generic Programming

Presenters:

David Richardson
Werner Krandick
Drexel University

A computer algebra library of C-programs, saclib, is improved using generic programming techniques such as concept design and template meta-programming. A new concept and a novel application of an existing STL concept is presented along with an implementation in C++ to allow the compiler to enforce new types of memory semantics. The implementation exposes existing memory leaks, improves program understanding, and facilitates safe programming practices that avoid memory leaks. These improvements require translating saclib from C to C++; almost all of this is done automatically. The new concepts and implementations are compared to existing practice in the Boost and STL libraries. A tool for the detection of memory leaks, Valgrind, is used to validate the removal of memory leaks. Timing experiments are performed to verify that no overhead cost is incurred at runtime.

29 Poster Title:

TeX/LaTeX to MathML Conversion: State of Affairs

Presenters:

Igor Rodionov
Stephen Watt
Ontario Research Centre for Computer Algebra (ORCCA) University of Western Ontario

TeX is easily the most widely used typesetting tool in use in the scientific community these days (and has been for more than a decade), particularly due to its superb handling of mathematics. Unfortunately, it is not an easy task to publish a TeX document on the Web, especially in such a way that it would look as good as was intended by its author, as well as be searchable. A few years ago (in 1999) solution to this problem became possible with the first version of MathML Mathematical Markup Language, which is an application of XML. Developed under the auspices of W3C, MathML made it possible to represent both presentational and semantic aspects of mathematical objects (formulae) in such a way that they could be easily included into larger XML and HTML documents. Quickly adopted and incorporated into a number of large mathematical software packages (e.g. Maple and Mathematica), it became obvious that translation of TeX-typeset formulae into MathML is useful not only for searching and publishing on the Web, but also for math communication.

For a number of reasons (that will be discussed), translation of TeX-encoded mathematics into MathML is anything but trivial. A number of attempts were made to tackle the problem, some more, some less successful, but none of them produced the results that would be good enough to be called completely satisfactory. As a result, here at ORCCA we undertook the task of creating a tool for translating TeX/LaTeX-encoded math into MathML. The result is an application (as well as our online service based on it) that we shall present here at ECCAD. Live demonstration will be given.

30 Poster Title:

Architecture-Aware Taylor Shift by 1

Presenter:

Anatole D. Ruslanov
Drexel University

Given an integer polynomial $A(x)$, Taylor shift by 1 computes the polynomial $B(x) = A(x+1)$. This operation is the most time-consuming subalgorithm of the Descartes method for polynomial real root isolation.

The classical implementation of the operation performs $\frac{n(n+1)}{2}$ integer additions where n is the degree of A. In fact, most existing implementations simply make calls to an integer addition routine, often to the GNU-MP routine for integer addition.

This holds for the computer algebra system Maple, for Hanrot's implementation of the Descartes method, and for von zur Gathen and Gerhard's implementation of the classical Taylor shift by 1. By contrast, the saclib library of computer algebra programs uses a specialized routine that does not make any function calls.

Our method is further specialized to take advantage of current processor architectures. The method utilizes instruction-level parallelism and efficiently handles the memory hierarchy. Unlike the GMP-package, our implementation does not rely on fine-tuning assembly language. Instead, our implementation can be adjusted automatically to a particular system by restructuring the code at a high-level and relying on features of the machine-specific compiler. By using various tiling schemes and delayed carry computation our implementation dramatically reduces the number of read instructions and cache misses. In addition, we obtain improved pipeline performance and better instruction-level parallelism.

The poster presents our method and compares to GMP-based Taylor shift and the saclib method.

31 Poster Title:

Tighter Probability Bounds for Randomized Linear Algebra Algorithms

Presenters:

B. David. Saunders
Zhendong Wan
University of Delaware

Many probabilistic algorithms for linear algebra computations over a finite field require choice of random matrices in order to achieve high likelihood of certain conditions. Most probabilistic estimates are based on a uniform distribution of elements from the field. For the modestly non-uniform distributions, We often handle it by using the Schwartz-Zippel lemma, or using a bound such as the number of success cases divided by the number of total cases times the minimal probability of success for any individual case. Are such probability bounds tight? No! We offer a better approach to some estimates. Consequently, tighter bounds will reduce the random size and the number of iterations to achieve a given probability of success. We show some application in linear algebra.

32 Poster Title:

Over-constrained Weierstrass iterations and the nearest consistent systems

Presenters:

Mark Sciabica
Agnes Szanto
North Carolina State University

We propose a generalization of the Weierstrass iteration for over-constrained systems of equations and we prove that the proposed method allows to find the nearest perturbed system with at least k common roots. We allow perturbations from some fixed finite dimensional vector space. In the univariate case we show the connection of our method to the optimization problem formulated by Karmarkar and Lakshman for the nearest GCD. In the multivariate case we generalize the expressions of Karmarkar and Lakshman, and give a simple iterative method to compute the optimum.

This work is a collaboration with Olivier Ruatta.

33 Poster Title:

Using Computer Algebra Systems In The Development of Mathematical Web Services

Presenters:

Elena Smirnova
Stephen Watt
Ontario Research Centre for Computer Algebra (ORCCA) University of Western Ontario

This presentation includes materials of join European-Canadian project "Mathematics on the Net" (MONET, ESPRIT-CANARIE).

1. Introduction

The MONET project is an investigation into distributed mathematical web services. The project is undertaken by the MONET consortium, consisting of several universities and institutions in Europe, and The University of Western Ontario in Canada(UWO). The aim of the MONET project is to create a framework of various mathematical services, accessible through the web as well as provide an engine resolving queries from the clients and finding an appropriate mathematical service for solving the particular problem.

The main idea of the MONET architecture allows a client to perform mathematical computations of various level of sophistication and does not require knowledge about special mathematical software used for these computations. The MONET approach let the user to access a wide variety of software packages to perform non-trivial mathematical computations as a part of their tasks even if appropriate mathematical software is not installed on user's computer or local network. Besides, it makes less sense to buy a full mathematical software package to solve one particular problem in certain field than to call a convenient mathematical web service.

One of the main challenges of the project is to provide effective and sophisticated algorithms to match the characteristics of a problem to the advertised capabilities of available services and then invoking the chosen services through a standard mechanism.

2. Architecture overview

MONET architecture includes three main components: 1) Client - a person or software package, requesting to solve a certain mathematical problem (for example a system of differential equations). 2) Mathematical service - a web service, based as usually on some powerful mathematical software package (for example Maple, NAG Numerical Libraries, Axiom, Aldor) advertising a number of mathematical problems that can be solved by this service. 3) Broker - the matching and planning engine that compare the problem description from user request to one offered by some of mathematical services.

All components of the architecture are hooked up together over the world network to carry out the full functionalities of distributed engine for solving of mathematical problems. For communication their use standard network protocols, OpenMath as well as specially designed XML-based languages and ontologies.

3. Design and Implementation of Mathematical Symbolic Services

The main assignment of UWO team in the MONET project was in developing of a Mathematical Symbolic Service Environment, which allows to solve various mathematical problems using symbolic computation approach.

3.1 Design

MAPLE was chosen to be the main solving engine, used by the current version of UWO mathematical services. Each mathematical web service is represented by a configuration file, which is created by the author of the service and describes interface of the service (using brand-new service description language) and service implementation (using language of solving software, in our case Maple). Configuration files are stored in XML format and can contain information about one or more math services. The configuration file has the following structure:

```
<mathServer>
    <msdl>
        <service name=\"sevice_A\">
              {complete service description,
              using Mathematical Service
              Description Languge(MSDL)
              for a service A}
        </service>
        {MSDL for other sevices}
    </msdl>

    <services>
        <service name=\"service_A\"
        call=\"function_call_for_service_A\"/>
           {interface for other services}
        </services>
    <implementation language = \"maple\"}
        {Maple implementation for each service}
    </implementation>
</mathSever>
```

This approach requires from the author of the service no knowledges about XML, java or web service technologies, but only about Maple!

3.2 Implementation

Implementation part includes various technologies and software tools to create and maintain mathematical symbolic Solver Environment. Among them: Java libraries, system shell scripts, XSLT-stylesheets, Maple packages to convert between Maple and OpenMath, JavaServer Pages, etc.

The developed software packages are designed to provide administrators of Symbolic Services with easy and efficient tools for creating new mathematical symbolic services and managing existing ones.

The software is downloadable from `http://www.orcca.on.ca/MONET/downloads/`.

4. Status

Currently more than 10 symbolic services are deployed on the UWO Symbolic Web Server. The remote clients for these services are available at `http://ptibonum.scl.csd.uwo.ca:16661/MonetServiceClient/`

34 Poster Title:

An Extensible OpenMath-Maple Translator

Presenters:

Clare So
Sandy Huerter
Stephen Watt
Ontario Research Centre for Computer Algebra (ORCCA) University of Western Ontario

OpenMath is an extensible, emerging, platform neutral standard to represent and exchange the semantics of advanced mathematics over the Internet. To use Maple as the computation engine in the MONET (Mathematics on the Net) web services, it is necessary for Maple to understand OpenMath inputs and generate OpenMath output. At present, Maple does not support OpenMath. Translating from OpenMath to Maple does not involve resolving ambiguity, but handling differences in the shapes of mathematical objects between the formats are needed. The reverse direction involves both resolving ambiguity and handling differences in the shapes of mathematical objects. This poster presents two translators developed in our laboratory for conversion between Maple and OpenMath. These translators are extensible in the sense that they can handle concepts from any collection of content dictionaries.

35 Poster Title:

A Brief Tour of MultInt: A Maple Package for Multiple Integration

Presenter:

Akalu Tefera
Grand Valley State University

Most identities in mathematics are usually hard to prove and often require lengthy and tedious verification. One of the most exciting discoveries in recent years, due to Herb Wilf and Doron Zeilberger (WZ), is that every *proper-hyperexponential* multi-integral identity with a *fixed* number of integration signs possesses a computer-contructible proof.

In general, the "objects" of study in the WZ theory are expressions of the kind

$$\sum_{\mathbf{k}} \int F(\mathbf{n}; \mathbf{k}, \mathbf{x}) d\mathbf{x}$$

and identities between them. In the above general integral-sum, $\mathbf{k}$, $\mathbf{n}$ are *discrete* multi-variables, while $\mathbf{x}$ is a *continuous* multi-variable, and F is *hyperexponential* in all its arguments.

The amazing discovery of Wilf and Zeilbeger was a general algorithm that produces a proof of an identity of the form

$$\sum_{\mathbf{k}} \int F(\mathbf{n}; \mathbf{k}, \mathbf{x}) d\mathbf{x} = \text{answer}(\mathbf{n})$$

and allows us to discover *new identities* whenever it succeeds in finding a proof certificate for a known one.

Presently the computer implementation of the WZ method is done by considering two special cases of the general integral-sum. One is the case of the pure *multisum*, i.e. $\mathbf{x}$ is empty, and the other is the case of the pure *multi-integral*, i.e. $\mathbf{k}$ is empty. Several implementation have been developed in Maple for the case of sum or multisum, for example the **sumtools** package.

In this poster we describe `MultInt`, a Maple implementation of the continuous version of the WZ method for symbolic evaluation of multiple integrals with application to computer generated proof of integral identities.

36 Poster Title:

Efficient runtime representations of domains in Aldor for interoperability with computer algebra systems

Presenter:

Geoff Wozniak
University of Western Ontario

Aldor is a language well suited for high performance algebraic computation. It is a language where types are first-class objects (i.e., they can be manipulated at runtime) in addition to being strongly typed, it can be compiled to native machine code, and is supported on a variety of modern platforms. However, development time tends to be longer with Aldor as opposed to other computer algebra systems (e.g., Maple) and as a result, lends itself to being a back-end for computation in other computer algebra systems.

With the goal of such interoperability in mind, problems with types may arise. Specifically, integrating a computer algebra system that is weakly typed with Aldor, which is strongly typed. To effect this synergy, it is worthwhile for the runtime system of Aldor to contain certain functionality that facilitates a simpler implementation from the perspective of the computer algebra system. Mainly, this includes reflective features for runtime type checking and more thorough operations for exploring the relationship between domains and categories. For example, the computer algebra system in question may call an Aldor function that returns an Aldor domain object. It is useful to be able to find out what exports are contained in said domain. Currently, this is achievable, but it is not easily realised. In this presentation, approaches and analysis are given that would allow for such functionality in a more simplistic fashion.

37 Poster Title:

An Experimental Handwritten Mathematical Symbol Reognition System

Presenter:

Xiaofang Xie
University of Western Ontario

The recognition for handwritten mathematical symbols plays an important role in computer algebra systems, such as Maple and Mathematica. The reasons are: the interface of handwriting can encourage more people use the computer algebra system than the traditional interface of typing characters; recognized handwritten symbols can be easily transfered to other formats, such as MathML, which is easy for editing and maintaining. With the development of hardwares, such as PDA, tablet PC, people can write directly on these devices. Therefore, recognition on handwritten mathematical symbols attracts more and more research interests. We described an experimental recognition system using Hidden Markov Model. The system contains four major modules: preprocessing module, feature extraction module, vector quantization module and Hidden Markov Model. In preprocessing module, we re-sampled the data in order to get rid of the factor of writing speed. We also smoothed and deslanted the data in this module. Feature extraction module is the key in handwriting recognition. We studied different kinds of features and applied some of them in this experimental recognition system. We used LBG VQ design algorithm for vector quantization and used Discrete Hidden Markov Model for recognition.

38 Poster Title:

Multiplication of Blackbox Matrices

Presenter:

George Yuhasz
North Carolina State University

Work done with Erich Kaltofen.

The multiplication of blackbox matrices, while trivial in the mathematical sense, creates interesting design and implementation issues when writing a generic blackbox library. I will present a solution to the multiplication of blackbox matrices, based on the blackbox archetype currently used in LinBox, a generic blackbox linear algebra library.

39 Poster Title:

Non-commutative Riquier Theory in Moving Frames of Differential Operators

Presenter:

Yang Zhang
University of Western Ontario

Moving frames chosen to be invariant under a known Lie group $\mathcal{G}$ provide a powerful generalization of the idea of choosing $\mathcal{G}$-invariant coordinates to cases where $\mathcal{G}$-invariant coordinates do not exist. Such $\mathcal{G}$-invariant formulations are of great current interest in areas such as Geometric Integration where $\mathcal{G}$-invariant integrators (e.g. symplectic integrators), can often substantially outperform non-invariant integrators. They are also of substantial interest in applications where one would like to factor out a known group.

One form of classical existence and uniqueness theory for analytic PDE referred to (standard) commuting partial derivatives is that of Riquier, which was formulated and generalized by Rust using a Gröbner style development.

We extend the Rust-Riquier existence and uniqueness theory to analytic PDE written in terms of moving frames of non-commuting Partial Differential Operators (PDO). The main idea for the theoretical development is to use the commutation relations between the PDO to place them in a standard order. This normalization is exploited to generalize the corresponding steps of the commuting Rust-Riquier Theory to the non-commuting case.

Given an equivalence group $\mathcal{G}$ Lisle has given a $\mathcal{G}$-invariant method for determining the structure of Lie symmetry groups of classes of PDE. Lisle's method for such group classification problems was illustrated on a number of challenging examples, which lead to unmanageable expression explosion for computer algebra programs using the standard (commuting) frame. He obtained new results, which for want of an existence and uniqueness theorem for PDE in non-commuting frames, had to be individually checked. We provide an existence and uniqueness theorem making rigorous the output from Lisle's method. For the finite parameter group case, the output is reformulated in terms of the integration of a system of Frobenius type, which can be numerically integrated by integrating an ODE system along a curve.

40 Poster Title:

An Algebraic Method for Analyzing Multibody Dynamic Systems

Presenter:

Wenqin Zhou
University of Western Ontario

In this poster we mainly consider the analysis of mechanical systems by combining the software package Dynaflex with RifSimp in Maple. Dynaflex generates the governing equations for mechanical systems and the RifSimp package is used for the symbolic analysis of differential equations. We show that the output equations from Dynaflex can be converted into a form in which they can be analyzed by RifSimp. Of particular interest is the ability of RifSimp to split a set of differential equations into different cases; for each case there are different assumptions made which can lead to significant simplifications. In order to allow RifSimp to conduct its analysis, the governing equations must be converted from a trigonometric form into a polynomial form. After this is done, RifSimp can analyze the system and present its results either graphically, or in list form.

We will give some easy examples, like 3D Spinning Tops, to explain how to use Dynaflex to get the analytical equations for the mechanical system. After that, we will see how to use RifSimp package to get all analytic cases to the mechanical system. Also an algorithm will be given for automatic simplification of the symbolic models with all the cases and RifSimp tree. For full cases, some is very simple, some maybe quite complicated, we can choose the simpler cases to analytically solve them and to complicated cases, we can numeric solve them since they are all in the canonical differential equation form. An important option with RifSimp is the possibility of excluding special cases that are known to be not of interest. Thus if RifSimp detects a special case, say $m = 0$, but we know that $m <> 0$, then we can pass this information to RifSimp in an optional list. Then we numeric or symbolic solve them. We conclude with some advantage and disadvantage of symbolic simplification method and point out some future works.

∴

MOCAA'2004 Invited Talk Abstracts

(in alphabetical order)

1

Speaker:

Edgardo Cheb-Terrab
Maplesoft

Title:

Computing Traveling Wave Solutions for non-linear autonomous PDE systems

Given a non-linear autonomous PDE system in unknowns $fi(xj)$, a traveling wave solution (TWS) is an exact closed form solution of the form

$$fi(\tau) = \sum_{k=0}^{n_i} A_{i,k}\tau^k$$

where the n_i are finite, the ${}_{i,k}$ are constants with respect to the ${}_j$, and $\tau = F\left(\sum k = 0^j c_k x_k\right)$ where the c_k are constants and F is typically a trigonometric or Jacobi elliptic function. This type of solution plays an important role in the study of non-linear physical phenomena. In this talk, the way TWS are constructed is reviewed, the new implementation in the Maple 9.5 system, together with a prototyping extension of it, are shown, and a generalization of the method taking t as the solution of a generic second order linear ODE is presented.

2

Speaker:

Robert Corless
ORCCA, University of Western Ontario

Title:

Whats "nu" about the derivative

Differentiation is an old topic, both mathematically and computationally, dating back at least to Fermat and Descartes, and possibly even to Archimedes. In more recent times, the first computer programs to differentiate expressions were written by Kahrimanian at Temple University and Nolan at MIT in 1953. [For comparison, work on Fortran started at IBM in 1954, with completion (of FORTRAN0) in 1957.] Why are we still working on computer differentiation, fifty years later?

There are in fact many reasons, and there remains a lot left to do, especially in the more recent topic of program or automatic differentiation. However, this talk concerns itself with something different. For an example of just what I mean, ask Maple to compute "diff(sin(x), x$nu)", for a symbolic "nu"; at present, you simply get your question returned, unanswered. In this talk I examine two possible meanings for this question, and show how to get answers, genuinely extending the power of Maple.

This is joint work with Mhenni M. Benghorbal, and benefited from the programming assistance of Ryan Morris.

3

Speaker:

John May
North Carolina State University

Title:

Approximate Factorization of Noisy Multivariate Polynomials

The input to our algorithm is a "noisy" polynomial $f(x, y)$, this is, its complex rational coefficients are considered imprecise with an unknown error that causes f to be irreducible over the complex numbers C. We seek to perturb the coefficients by a small quantity such that the resulting polynomial factors over C. Ideally, one would like to minimize the perturbation in some selected distance measure, but no efficient algorithm for that is known. We present a numerical multivariate greatest common divisor algorithm and use it for a numerical variant of algorithms by W. M. Ruppert and S. Gao. Our numerical factorizer makes repeated use of singular value decompositions. We demonstrate on a significant body of experimental data that our algorithm is practical and can find factorizable polynomials within a distance that is about the same in relative magnitude as the input error, that even when the relative error in the input is substantial (10^{-2}).

Joint work with Shuhong Gao (Clemson), Erich Kaltofen (NCSU), Zhengfeng Yang and Lihong Zhi (AMSS Beijing)

4

Speaker:

Michael Monagan
CECM, Simon Fraser University

Title:

Polynomial Gcd Computation

In the early days of computer algebra, the lack of an efficient solution to the problem of computing the GCD of two multivariate polynomials over the integers was the main problem that greatly restricted the general problem solving capability of CA systems. The breakthrough came with Brown's modular GCD algorithm in 1971. Subsequent work in the 1970's, 80's and early 90's by many authors led to improved efficiency for polynomial GCD computation over the integers, finite fields, and number fields. Yet the problem of intermediate expression swell remains; polynomial GCD computation over more complicated rings and fields is done using non modular methods and this means that computer algebra systems are very limited in the size of problems they can work with.

In recent work we have extended the modular GCD algorithm to work over an algebraic function field in one or more parameters and also to work when the defining polynomial of the function field is not irreducible. As a consequence we obtain an algorithm for multivariate GCD computation over algebraic number fields and functions fields via moving polynomial variables into the coefficient field.

In this talk I will present the key ideas of Brown's algorithm, Encarnacion's output sensitive algorithm for number fields, our new algorithm for function fields, and also the non-modular algorithm of Moreno Maza and Rioboo for univariate GCD computation modulo a triangular set. I will give some Maple timings to indicate the performance improvement we are seeing when comparing our modular algorithm with the non-modular algorithm of Moreno Maza and Rioboo and with a "primitive" version of their algorithm.

This is joint work with Mark van Hoeij of Florida State University.

5

Speaker:

Gilles Villard
CNRS, ENS-Lyon

Title:

Lattice-Based Memory Allocation

We investigate the problem of memory reuse, with the goal of reducing the necessary memory size for an array variable, in the context of synthesis of dedicated processors or compilation. Memory reuse is a well-known concept when allocating registers (i.e., scalar variables). Its (recent) extension to arrays was studied mainly by Lefebvre and Feautrier (for loop parallelization) and by Quilleri and Rajopadhye (for circuit synthesis based on recurrence equations). We develop a mathematical framework based on (integral) critical lattices that subsumes all previous approaches and gives new insights into the problem. Through the notion of conflicting indices we define an integral lattice, admissible for the set of differences of conflicting indices, used to build a valid linear allocation. At the center of our approach is the constrained choice of a certain lattice whose determinant gives the storage requirement for the array in question. We place the problem in context, showing its relation to critical lattices, successive minima, and basis reduction, and we analyze various strategies for lattice-based memory allocation.

This work has been done in collaboration with Alain Darte (CNRS/LIP ENS-Lyon) and Rob Schreiber (Hewlett Packard Laboratories, Palo Alto).

∴

MOCAA'2004
Contributed Talk Abstracts
(in alphabetical order)

6

Speaker:

Amir Amiraslani
University of Western Ontario

Title:

Some Aspects of Polynomial Algebra by Values

This talk outlines a new algorithm for finding the roots of multi-variate polynomials using the division algorithm. The point of view taken in the talk is that all polynomials considered are given by their values at known points and that the degrees of the polynomials are known or can be deduced. We choose to perform all operations directly on the given values in order to avoid potential numerical instability in changing bases; of course this prevents us from discovering and using any possible sparse representation, which might be cheaper, but our goal is to provide stable algorithms.

7

Speaker:

Mhenni Benghorbal
University of Western Ontario

Title:

Arbitrary Order Symbolic Derivatives and Integrals

We discuss methods for finding the nth derivative of certain classes of functions. We recall the rational function approach of full partial fraction decomposition, and extend this by the Mittag-Leffler theorem to meromorphic functions whose residues can be calculated symbolically. We also discuss finding the nth derivative of power-exponential functions by explicit construction and solution of recurrence equations for the derivatives. These computations are useful for finding formal power series, for finding the solutions to certain fractional differential equations, for symbolic integration, and also lead to some new functional identities.

This is a joint work with Dr. Robert Corless

8

Speaker:

Reinhold Burger
University of Waterloo

Title:

Closed form solutions of linear ODEs having elliptic function coefficients

We consider the problem of finding closed form solutions of linear differential equations having coefficients which are elliptic functions. For second order equations we show how to solve such an ode in terms of doubly periodic functions of the second kind. The method depends on two procedures, the first using a second symmetric power of an ode along with a decision procedure for determining when such equations have elliptic function solutions, while the second uses computation of exponential solutions.

This is joint work with George Labahn (University of Waterloo) and Mark van Hoeij (Florida State University).

9

Speaker:

Jacques Carette
McMaster University

Title:

A new normal form algorithm for piecewise functions

By using the underlying linear order inherent in the real numbers, we create a new data-structure representation for piecewise functions. This representation allows normalization to happen using a linear number of steps (with respect to the number of breaks); previous work needed exponentially many steps. The representation also allows some computations to proceed with partially defined piecewise functions, where previous work needed total functions.

10

Speaker:

Howard Cheng
University of Lethbridge

Title:

Output-sensitive Modular Algorithms for Polynomial Matrix Normal Forms

We give modular algorithms to compute a row-reduced form and a weak Popov form of a polynomial matrix, improving on existing fraction-free algorithms. In each case we define lucky homomorphisms, determine the appropriate normalization, as well as bound the number of homomorphic images required. The algorithms have the advantage that they are output-sensitive, that is, the number of homomorphic images required depends on the size of the output. Furthermore, there is no need to verify the result by trial division or multiplication.

Finally, our algorithms can be used to compute normalized one-sided greatest common divisors and least common multiples of polynomial matrices along with irreducible matrix-fraction descriptions of matrix rational functions.

This work was done jointly with George Labahn (University of Waterloo).

11

Speaker:

Jennifer de Kleine
Simon Fraser University

Title:

Zippel's Algorithm Extended for the Non-Monic Case

The sparse modular GCD algorithm was presented by Zippel for computing the greatest common divisor of two multivariate polynomials over the integers. We extend this algorithm to work for non-monic GCDs. The non-monic case occurs when the GCD has a leading coefficient involving one or more variables. For example,the GCD $G = (4y + 2z)x^2 + 7$ in $Z[x, y, z]$ is non-monic in the main variable x. The problem is that at the bottom level of the algorithm we call the Euclidean algorithm over the integers modulo p, p a prime, which returns a monic GCD. We describe various approaches for handling the non-monic case, in particular, a normalization technique and a method which uses a sparse rational function interpolation algorithm.

12

Speaker:

Wayne Eberly
University of Calgary

Title:

Reliable Black Box Algorithms for Matrix Nullspace and Rank

Black box algorithms to solve singular systems of equations or sample from the nullspace are generally unreliable unless the input matrix is conditioned in some way. A variety of efficient conditioners are now known and provably reliable for computations over large fields. However, the choice is much more limited for small field computations.

In this talk, properties of a "sparse conditioner" for computations over small fields will be discussed. It will be shown that this conditioner can be used in an algorithm for computation of the rank of a matrix, over a small field, that is asymptotically more efficient than previously available algorithms.

If time permits, block Lanczos algorithms will also be discussed. In particular, a block Lanczos algorithm that is probably reliable for computations over large fields, and considerably simpler than those that are currently described in the literature and implemented, will be discussed.

13

Speaker:

Ronald Ferguson
Simon Fraser University

Title:

Computation of Mahler's Measure

The Mahler's measure of a monic polynomial is the product of the moduli of all the roots of the polynomial on or outside the unit circle. For reciprocal polynomial with small Mahler's measure we discuss an efficient algorithm for locating these roots.

This is joint work with Peter Borwein.

14

Speaker:

Jeremy Johnson
Drexel University

Title:

Performance Models for the Walsh-Hadamard Transform

I will present performance models that take into account instruction count, register spills, and cache misses for a family of algorithms to compute the Walsh-Hadamard Transform (WHT). These algorithms are related to the FFT, and the attention drawn to these aspects of performance may be useful for a wide range of algorithms. The performance models were implemented and analyzed using Maple.

15

Speaker:

Ilias Kotsireas
Wilfrid Laurier University

Title:

Genetic Algorithms and Computer Algebra

Genetic Algorithms are an algorithmic paradigm which mimics biological processes that occur in the theory of evolution. Genetic Algorithms have a very wide range of applicability and they are a powerful alternative to the more conventional search algorithms. The mechanics of Genetic Algorithms will be illustrated with a particular application in Combinatorics using Computer Algebra and high-performance computing.

16

Speaker:

Kaska Kowalska
McMaster University

Title:

Synthesis of Model Predictive Controllers With the Help of Symbolic Computation

This talk will introduce a symbolic framework used for the design of model predictive controllers. The framework employs a computer algebra system to produce symbolic, closed form solutions of the model of the plant. These parameterized closed form solutions are then used to predict the state of the system. These predictions allow the real time control algorithm to rank the different available control actions and make a decision about the next optimal control move. One of the novelties of the presented approach includes a reduction in the run-time complexity and an increase in the accuracy of the solution. The presented framework also incorporates automatic code generation. The talk will focus on the use of symbolic methods for controller synthesis and will discuss some of the challenges associated with building a framework with computational algebra tools.

17

Speaker:

Marc Moreno Maza
University of Western Ontario

Title:

On Polynomial Gcds over Direct Products of Fields

Let K be a field of multivariate rational functions over an infinite field. Let L be a direct product of fields extending K by a tower of simple algebraic extensions. We present a modular algorithm for computing polynomial gcds over L based on a Hensel lifting strategy.

The rational reconstruction is avoided by "guessing" the denominator of the gcd before the lifting step. The algorithm is probabilistic but succeeds with probability one. Our implementation shows a significant improvement with respect to other methods.

This is joint work with Cosmin Oancea (University of Western Ontario) and Francois Boulier (Universite de Lille 1).

18

Speaker:

Ned Nedialkov
McMaster University

Title:

Solving Differential-Algebraic Equations by Taylor Series

We present a method for solving numerically an initial-value problem differential algebraic equation (DAE). The DAE can be of high-index, fully implicit, and contain derivatives of order higher than one.

We do not reduce a DAE to a first-order, lower-index form: we solve it directly by expanding its solution in Taylor series. To compute Taylor coefficients, we employ the structural analysis of J. Pryce and automatic differentiation.

Generally, our approach succeeds for any DAE whose sparsity structure correctly represents its mathematical structure. We show that a failure occurs if and only if the "system Jacobian" of the DAE is structurally singular up to roundoff, a situation recognizable in practice.

This method has been implemented in a C++ code by N. Nedialkov. Numerical results on several standard test problems show that it is both efficient and accurate.

This is a joint work with J. Pryce, Royal Military College of Science, England.

19

Speaker:

Victor Pan
Lehman College, CUNY

Title:

Root-finding and Eigen-solving

By using a matrix approach to polynomial root-finding, we have devised a QR-like algorithm that uses linear space and linear time per iteration, while at the same time preserving the robustness and rapid convergence of the classical QR algorithm.

This is joint work with Dario Bini and Luca Gemignani

20

Speaker:

Roman Pearce
Simon Fraser University

Title:

Computing in Polynomial Quotient Rings

We present algorithms for polynomial division, factorization, and rational expression simplification modulo an ideal of the polynomial ring. Some familiarity with Groebner bases is assumed.

21

Speaker:

Agnes Szanto
North Carolina State University

Title:

Over-constrained Weierstrass iteration and the nearest consistent system

We propose a generalization of the Weierstrass iteration for over-constrained systems of equations and we prove that the proposed method allows us to find the nearest system which has at least k common roots and which is obtained via a perturbation of prescribed structure. In the univariate case we show the connection of our method to the optimization problem formulated by Karmarkar and Lakshman for the nearest GCD. In the multivariate case we generalize the expressions of Karmarkar and Lakshman, and give a simple iterative method to compute the optimum.

22

Speaker:

Thomas Wolf
Brock University

Title:

Towards a Classification of Supersymmetric Evolutionary PDE

The talk starts with explaining the concept of symmetries of evolutionary partial differential equations (PDEs) and of supersymmetry of PDEs. The problem of finding polynomial supersymmetric evolutionary PDEs leads to a bi-linear algebraic systems to be solved. Investigating fermionic and bosonic systems of different differential order, different order of the symmetry and different weightings of the unknows results in too many algebraic systems to be solve interactively. The safety and other issues of the automatic solution of such systems is addressed in the second part of the talk.

∴

ECCAD'2004/MOCAA'2004 List of Participants

1. Kamal Abdali, National Science Foundation
2. Michael Abramson, National Security Agency
3. Iyad A. Ajwa, Ashland University
4. Amir Amiraslani, University of Western Ontario
5. Dhavide Aruliah, University of Western Ontario
6. Jan Bakus, Maplesoft
7. Mhenni Benghorbal, University of Western Ontario
8. Jonathan M. Borwein, Dalhousie University
9. Chris Brown, United States Naval Academy
10. Reinhold Burger, University of Waterloo
11. Dan Butcher, Wilfrid Laurier University
12. Jinlong Cai, University of Western Ontario
13. Colin Campbell, TechnicalMastery.com
14. Jacques Carette, McMaster University
15. Edgardo Cheb-Terrab, Maplesoft
16. Zhuliang Chen, University of Waterloo
17. Howard Cheng, University of Lethbridge
18. Michael Cherkassoff Maplesoft
19. Paulina Chin, Maplesoft
20. Arthur D. Chtcherba, University Of Texas - Pan American
21. Robert M. Corless, University of Western Ontario
22. Jason Cousineau, Wilfrid Laurier University
23. David A. Cox, Amherst College
24. Timothy Daly, City College of New York
25. Jennifer de Kleine, Simon Fraser University
26. Hui Ding, University of Western Ontario
27. Laurentiu Dragan, University of Western Ontario
28. Kevin Durdle, University of Western Ontario
29. Wayne Eberly, University of Calgary
30. Jeff Farr, Simon Fraser University
31. Ronald Ferguson, Simon Fraser University
32. Stephen Forrest, Maplesoft
33. Marie-Paule Gagne-Portelance, University of Western Ontario
34. Keith Geddes, University of Waterloo
35. Jürgen Gerhard, Maplesoft
36. Mark Giesbrecht, University of Waterloo
37. Marshall Hampton, University of Minnesota
38. Dave Hare, Maplesoft
39. Jason Hinek, University of Waterloo
40. Bradford Hovinen, University of Waterloo
41. Sandy Huerter, University of Western Ontario
42. Silvana Ilie, University of Western Ontario
43. David Jeffrey, University of Western Ontario
44. Jeremy Johnson, Drexel University
45. Erich Kaltofen, North Carolina State University
46. Ilias S. Kotsireas, Wilfrid Laurier University
47. Kaska Kowalska, McMaster University
48. George Labahn, University of Waterloo
49. Edmond Lau, maximalgorithms.com
50. Daniel Lazard, Université Paris 6
51. Desmond Leung, Wilfrid Laurier University
52. David Linder, Maplesoft
53. Michael Lloyd, University of Western Ontario
54. Austin Lobo, Washington College
55. Paul Mansfield, Maplesoft
56. John May, North Carolina State University
57. David Miller, McMaster University
58. Michael Monagan, Simon Fraser University
59. Marc Moreno Maza, University of Western Ontario
60. John M. Nahay, Swan Orchestral Systems
61. Ned Nedialkov, McMaster University
62. Cosmin Oancea, University of Western Ontario
63. Victor Pan, Lehman College, CUNY
64. Roman Pearce, Simon Fraser University
65. Scott R. Pope, North Carolina State University
66. David Richardson, Drexel University
67. Tom Robinson, University of Waterloo
68. Igor Rodionov, University of Western Ontario
69. Anatole D. Ruslanov, Drexel University
70. B. David Saunders, University of Delaware
71. Jason Schattman, Maplesoft
72. Mark Sciabica, North Carolina State University
73. Azar Shakoori, University of Western Ontario
74. William Sit, City College of New York
75. Elena Smirnova, University of Western Ontario
76. Clare So, University of Western Ontario
77. Alex Stewart, University of Waterloo
78. Arne Storjohann, University of Waterloo
79. Agnes Szanto, North Carolina State University
80. Akalu Tefera, Grand Valley State University
81. Gilles Villard, CNRS, ENS-Lyon
82. Zhendong Wan, University of Delaware
83. Stephen Watt, University of Western Ontario
84. Thomas Wolf, Brock University
85. Geoff Wozniak, University of Western Ontario
86. Xiaofang Xie, University of Western Ontario
87. Yuzhen Xie, University of Western Ontario
88. George Yuhasz, North Carolina State University
89. Yang Zhang, University of Western Ontario
90. Wei Zhou, University of Waterloo
91. Wenqin Zhou, University of Western Ontario
92. Eugene Zima, University of Waterloo

EAST COAST COMPUTER ALGEBRA DAY 2004 (ECCAD'2004)

I am very pleased on behalf of WLU to welcome ECCAD to Laurier for your 2004 meeting. Laurier is an institution that dates back to 1911 and has a long and proud tradition of providing accessible post-secondary education in Ontario.

Laurier in recent years has placed a renewed focus on Science with the creation of an independent Faculty of Science. With the Faculty of Science we are particularly proud of the work of our faculty in the Department of Computing who are your hosts for this meeting.

Once again, I welcome you to Laurier. This region of Waterloo is one of the most diverse and interesting in Canada and I would encourage you to take in some of our cultural and community activities during your visit to Waterloo.

Bob Rosehart

President

Wilfrid Laurier University

Message from Dr. Adele Reinhrartz, Dean of Graduate Studies and Research

The Faculty of Graduate Studies and Research welcomes ECCAD 2004 to Wilfrid Laurier University, and extends hearty congratulations to the Computer Algebra Research Group in Laurier's Department of Physics and Computer Science and the conference organizers. We are extremely proud of the important and high quality research being done at Laurier in this field, and we are confident that the day will be stimulating and productive, both in terms of the research developments presented, and in the interactions among scholars.

Best wishes for a successful and enjoyable conference

Adele Reinhartz, PhD
Dean of Graduate Studies and Research
Wilfrid Laurier University

Message from Dr. Arthur G. Szabo, Dean of Science

It is my pleasure to welcome you to the 2004 East Coast Computer Algebra Day on behalf of the Laurier's Faculty of Science. It is an honour for Laurier Science to be one of the sponsors for this important event.

Our Faculty consists of six departments i.e. Biology, Chemistry, Kinesiology & Physical Education, Mathematics, Physics & Computer Science, and Psychology. We are a young Faculty (formed in 2000 from a division of the Laurier's longstanding Faculty of Arts and Science), with a strong focus on research in the natural, physical, and social sciences. While you're here, do take note of our just-opened Laurier Science Research Centre, attached to the Science Building. The Centre will be the home of research enterprise in each of our disciplines, including Computer Science.

I congratulate the organizers of the ECCAD'2004 for the high quality of the program.
Welcome to the Laurier campus, and enjoy the meeting.

Arthur G. Szabo, PhD, FCIC
Dean of Science
Wilfrid Laurier University

Message from Maplesoft

Firstly, I would like to take this opportunity to thank the organizers of ECCAD 2004. Those of us who have been involved in organizing an event like this understand the work and dedication that is involved.

Maplesoft is very proud to be a sponsor of ECCAD 2004. Support of the computer algebra research community is key to our success, and we hope that our sponsorship demonstrates our commitment.

I hope all participants have a productive and enjoyable conference.

Dr. Laurent Bernardin
Vice President, Research and Development, Maplesoft

About Maplesoft

Maplesoft is a world leader in mathematical and analytical software. Its suite of products include Maple 9.5, the standard among interactive mathematical software, MapleNet a comprehensive mathematical infrastructure for the Web, and Maple T.A. a system for automated grading of mathematics tests. Maplesoft products embody the most advanced and integrated technology for both numeric and symbolic solution of complex problems. Its standards-compliant algorithms are renowned for is speed, accuracy and reliability. In addition, Maplesoft leads the market with innovations that make the management of technical knowledge – whether in industry, or in a classroom – more effective and efficient. Over 5 million users benefit from advanced Maple technology. Maplesoft's industrial customer base includes Boeing, Bosch, Canon, and NASA. Additionally virtually every major university and research institute in the world, including MIT, Stanford, Oxford and Waterloo, has adopted Maple products to enhance their education and research activities.
For more information visit www.maplesoft.com.

Maplesoft™
command the brilliance™

SHARCNET™

message from the executive director

carmen s. gicante

On behalf of SHARCNET, I would like to congratulate ECCAD'2004 organizers, sponsors and participants. SHARCNET is proud to support the world-class research of over 400 national and international research groups through the provision of high performance computing resources and services. SHARCNET exists to enable world-class computational research so as to accelerate the production of research results of benefit to the Canadian economy, our health, the environment and our general welfare.

Symbolic Computation is a crucial technology in many established and emerging industries and demonstrates outstanding promise for future innovation. SHARCNET is proud to sponsor events like ECCAD'2004 for precisely this reason.

Best wishes for a successful conference!

Carmen S. Gicante, P.Eng., MBA

about sharcnet

The Shared Hierarchical Academic Research Computing Network (SHARCNET) is a multi-institutional high performance computing (HPC) institute. A leading provider of HPC resources and services, SHARCNET accelerates the production of research results for some of Canada's pre-eminent academics; from increasing the understanding of outbreaks of diseases such as SARS, to the development of new models to manage financial risk. SHARCNET consists of a consortium of 11 geographically-distributed HPC clusters at academic institutions across South Central Ontario. The consortium is led by the University of Western Ontario, and includes the Universities of Guelph, McMaster, Wilfrid Laurier, Windsor, Waterloo, Brock, Ontario Institute of Technology and York, and colleges Fanshawe and Sheridan. SHARCNET is founded on academic-industrial collaboration. Its private sector partners include Hewlett Packard, Platform Computing, Bell Canada, Nortel Networks, Quadrics Ltd, and the Optical Regional Advanced Network of Ontario (ORANO). SHARCNET is generously supported by the Canada Foundation for Innovation, Ontario Innovation Trust, Ontario Research and Development Challenge Fund, as well as by its institutional members and industrial partners.

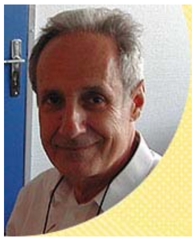
ECCAD'2004 Invited Speakers

Prof. Jonathan M. Borwein, FRSC
Dalhousie University, CANADA

Mathematics by Experiment: Plausible Reasoning in the 21st Century

Prof. David A. Cox
Amherst College, USA

Implicitization and Commutative Algebra



Prof. Daniel Lazard
Université Paris 6, FRANCE

Solving zero-dimensional systems of equations and inequations, depending on parameters

ECCAD'2004 Sponsors

- **Wilfrid Laurier University Research Office**
- **Wilfrid Laurier University Faculty of Science**
- **Maplesoft Inc.**
- **NSF**
- **SHARCNET**
- **NSERC**
- **ORCCA**
- **MITACS**
- **TechnicalMastery.com**

Getting the most from Maple's LinearAlgebra

May 8th, 2004

David Linder, Math Group Team Lead, Maplesoft

Efficient use of the datatype

Matrices and Vectors are examples of Maple's rtable data structure. Maple can store rtables with floating-point datatypes .

A datatype is a restrictive quality of the underlying structure. No element can be assigned to an entry if it is not of

the prescribed type. This provides the means for efficient implementation of numeric programs.

For datatype=float[8] the data is stored as a collection of hardware double-precision floating-point numbers.

For datatype=sfloat the data is stored as a collection of floating-point numbers, each in Maple's own software floating-point

datastructure.

Two principal purposes of these special datatypes are fast algorithmic selection and use in external calling. These are

Fast algorithmic selection

One writes an program to operate over Matrices and Vectors. Suppose that a special, efficient algorithm is

found which can be implemented only over a particular domain, say the integers.

A type-check such as `type(inputM,'Matrix(integer)')` is the means chosen to decide which path of the program to follow. How can this type-check work? If the datatype is anything then the check

entails walking the entire structure and checking each element in turn. That is costly, despite being made

faster in Maple 9.5 . If the datatype is integer then the type-check is immediate, it only has to check the rtable's

datatype.

Use in external calling.

Maple has compiled external functions available for linear algebra computation. These work directly with the data

portions of floating-point datatype Matrices and Vectors, and rely upon the data being stored in contiguous blocks

(ie, in arrays, in C). This is one of the principal motivations behind the existence of rtables! In order to avoid

unecessary duplication or copying of data, it is most efficient to use these entirely, if floating-point numeric linear

algebra is to be done.

Do not do the following, if possible, as it involves extraneous copies of the data. The computation of the LU factorization

will convert the Matrix to float8] datatype, in order to access the external routine.

```
> M := Matrix(100,100,(i,j)->1/j):
  M := evalf(M);
  infolevel[LinearAlgebra]:=2;
  M.M;
```

$$M := \begin{bmatrix} \text{100 x 100 Matrix} \\ \text{Data Type: anything} \\ \text{Storage: rectangular} \\ \text{Order: Fortran\_order} \end{bmatrix}$$

$$\text{infolevel}_{\text{LinearAlgebra}} := 2$$

```
MatrixMatrixMultiply:, "copying first Matrix to enable external call"
MatrixMatrixMultiply:, "copying second Matrix to enable external call"
MatrixMatrixMultiply:, "calling external function"
MatrixMatrixMultiply:, "NAG", hw_f06yaf
```

$$\begin{bmatrix} \text{100 x 100 Matrix} \\ \text{Data Type: float[8]} \\ \text{Storage: rectangular} \\ \text{Order: Fortran\_order} \end{bmatrix}$$

Instead, create the Matrix directly with the appropriate datatype.

```
> M := Matrix(100,100,(i,j)->1/j,datatype=float);
  M.M;
```

$$M := \begin{bmatrix} \text{100 x 100 Matrix} \\ \text{Data Type: float[8]} \\ \text{Storage: rectangular} \\ \text{Order: Fortran\_order} \end{bmatrix}$$

```
MatrixMatrixMultiply:, "calling external function"
MatrixMatrixMultiply:, "NAG", hw_f06yaf
```

$$\begin{bmatrix} \text{100 x 100 Matrix} \\ \text{Data Type: float[8]} \\ \text{Storage: rectangular} \\ \text{Order: Fortran\_order} \end{bmatrix}$$

Efficient use of memory

Maple uses contiguous blocks of its own allocated memory for the floating-point datatype rtables. Once a variable name

is reassigned the contiguous block of memory space used for that rtable is freed, yet the nearby surrounding memory

may be still used by something else.

In this initial scenario, a larger rtable always replaces a smaller one. Since the memory block used for the first rtable is

not large enough to hold the data of the second, and so on, and since the surrounding memory may be in use, then an

entirely new portion of allocated memory must be used. Maple can allocate more memory to use, but this situation is

not allowing it to re-use the smaller rtables' memory block for holding the new rtables.

In order to get the clearest picture of what can happen, examine the two cases below.

Each method creates Matrices of differing sizes, assigning each to the same variable. In the first case, the Matrices

are created, and disposed of, in order of ascending size. In the second case, they are created in order of descending

size. Notice how the total allocated memory used by Maple is much larger in the first case. Each subsequent Matrix

must be apportioned a new block of memory for its data, since the old block is not large enough to hold it. In the

second case, each of the subsequent Matrices data does fit into the previous one's memory block, so no new allocation takes place.

These operations should be performed in separate worksheets, to fully illustrate the memory savings. As an exercise,

do these computations separately. How does the total time differ, for each? How does the total memory allocation

differ for each? And the total memory used? Attempt to explain these differences and similarities.

```
> restart:
  (st,ba) := time(), kernelopts(bytesalloc):
  for i from 100 to 2000 by 100 do
     M := Matrix(i,i,datatype=float);
  od:
  time()-st, kernelopts(bytesalloc)-ba;
                                              0.540, 230120288
```

```
> restart:
  (st,ba) := time(), kernelopts(bytesalloc):
  for i from 2000 to 100 by -100 do
     M := Matrix(i,i,datatype=float);
  od:
  time()-st, kernelopts(bytesalloc)-ba;
                                              0.710, 32041236
```

Operations using NAG format output

In some situations, a compact form of the result can save memory. For example, the result of an LU factorization

can be contained in just a Vector and a single Matrix. The Vector represents the permutations done for best

pivoting, and the resulting Matrix contains, superimposed, the U and the L (with implicit unit-diagonal).

```
> restart:
  with(LinearAlgebra):
  M := RandomMatrix(3,outputoptions=[datatype=float]);
  (p,lu) := LUDecomposition(M,output=['NAG']);
  b := Vector(3,(i)->1,datatype=float):
  x := LinearSolve([p,lu],b):
  Norm( M . x - b );
```

$$M := \begin{bmatrix} -37. & 76. & -8. \\ -85. & -51. & 9. \\ -96. & 46. & 93. \end{bmatrix}$$

$$p, lu := \begin{bmatrix} 3 \\ 2 \\ 3 \end{bmatrix},$$

$$\begin{bmatrix} -96. & 46. & 93. \\ 0.885416666666666629 & -91.7291666666666714 & -73.3437500000000000 \\ 0.385416666666666629 & -0.635248694072223507 & -90.4352714058596376 \end{bmatrix}$$

$$1.11022302462515654\ 10^{-16}$$

Other operations may make use of this compact form, without having to recompute the $O(N^3)$ decomposition.

```
> infolevel[LinearAlgebra]:=2;
  ConditionNumber(M);
  ConditionNumber(M,factors=[p,lu]);
```

$$infolevel_{LinearAlgebra} := 2$$

```
ConditionNumber:, "calling external function"
ConditionNumber:, "NAG", hw_f06raf
ConditionNumber:, "NAG", hw_f07adf
ConditionNumber:, "NAG", hw_f07agf
```

$$6.712363426$$

```
ConditionNumber:, "calling external function"
ConditionNumber:, "NAG", hw_f06raf
ConditionNumber:, "NAG", hw_f07agf
```

$$6.712363426$$

```
> MatrixInverse(M);
  MatrixInverse([p,lu]);
MatrixInverse:, "calling external function"
MatrixInverse:, "NAG", hw_f07adf
MatrixInverse:, "NAG", hw_f07ajf
```

[[-0.00647560879135781771, -0.00933733313409670918, 0.000346571267483952902],
[0.00884133439983525449, -0.00528521182913031882, 0.00127201700710596584],
[-0.0110576325415351860, -0.00702434663154074840, 0.0104812694553933893]]

```
MatrixInverse:, "calling external function"
MatrixInverse:, "NAG", hw_f07ajf
```

```
[[-0.00647560879135781771, -0.00933733313409670918, 0.000346571267483952902],
    [0.00884133439983525449, -0.00528521182913031882, 0.00127201700710596584],
    [-0.0110576325415351860, -0.00702434663154074840, 0.0104812694553933893]]
```

Show how a linear system may be solved, repeatedly for several distinct RHS Vectors, while re-using the prefactored

components [p,lu]. As an additional exercise, show how this may be done with the QRDecomposition rouine.

Operations done in-place

Many routines within LinearAlgebra allow for in-place computation. The syntax is consistent, by providing

the option inplace = true .

inplace = true

```
> restart:
  M := LinearAlgebra:-RandomMatrix(3);
  N := LinearAlgebra:-RandomMatrix(3):
  with(LinearAlgebra):
  MatrixAdd(M,N,1,-3,inplace=true):
  M;
```

$$M := \begin{bmatrix} -21 & -50 & -79 \\ -56 & 30 & -71 \\ -8 & 62 & 28 \end{bmatrix}$$

$$\begin{bmatrix} -219 & -110 & 23 \\ 52 & 51 & 115 \\ 115 & 14 & 298 \end{bmatrix}$$

Routines which accept this option include LinearSolve, Add, Transpose, etc.

Some faster algorithms

Solving linear systems is one of the most typical operations in linear algebra. In Maple this can be done, as a floating-point computation, in any arbitrary extended precision. According to the earlier sections this can be done more efficiently by using the floating-point datatypes.

Here is some data, to express and example problem:

```
> Digits := 40:
  with(LinearAlgebra):
  M := RandomMatrix( 100, outputoptions=[datatype=float] ):
  b := RandomVector( 100, outputoptions=[datatype=float] ):
```

Prior to Maple 9 , the linear system $M \, . \, x = b$ would typically be solved as follows,by means of an LU factorization.

```
> time( assign('x',LinearSolve( M, b, method = LU )) );
                                  12.170
```

And one could perform one of several sorts of test on the answer.

```
> Norm( M . x - b );
```

$$4.441\ 10^{-35}$$

The above has be done entirely at a fixed precision of 100 digits. In particular, the $O(N^3)$ factorization has been done entirely with Maple's software floating-point numbers, using its internal kernel to do all the arithmetic operations.

The following is a new implementation which is done partly at hardware double-precision and partly in extended precision using software floating-point arithmetic. The operation is timed.

```
> time( assign('x',LinearSolve( M, b, method = hybrid)) );
                                  2.939
```

Again, one may test the answer by computing its forward-error.

```
> Norm( M . x - b );
```

$$1.041\ 10^{-35}$$

In this second computation, the crucial $O(N^3)$ factorization is done at fast non-interpreted hardware double-precision. This is followed by $O(N^2)$ iterative refinement of the intermediary solution at

the extended sortware precision. All these computations are done externally, for speed, using NAG or CLAPACK routines.

It should be possible to derive additional hybrid algorithms for other extended fixed precision computations,

by replacing some sections of highest complexity with a hardware precision operation and following that by extended precision refinement involving a lower complexity. Eigenvalue and eigenvector computation are likely candidates, as is multivariable Newton's method for rootfinding.

Augmenting Maple through external-calling

Suppose that you have an algorithm of your own, for some purely numeric computation in linear algebra. You

wish to implement it as efficiently as possible within Maple. You want it to be as fast as if you had implemented

in a compiled language, such as C. See the example worksheet on External Calling.

Numeric functionality new toMaple 9.5

There are a number of new packages and added functionality for numerics in Maple 9.5 .

Optimization

The Optimization package is entirely new to Maple 9.5. It provides fast arbitrary solution to the problems of floating-point local optimization. The solvers are arranged according to problem class. These classes include linear programming (LP), least squares (LS), and nonlinear programming (NLP).

The underlying implementations of the solvers are routines provided by NAG, as compiled C, and are accessed by Maple via the external-calling mechanism.

```
> Optimization:-Interactive();
                                        no solution available
```

Question: for what classes of problem will the optimal local extrema found by this package be global extrema?

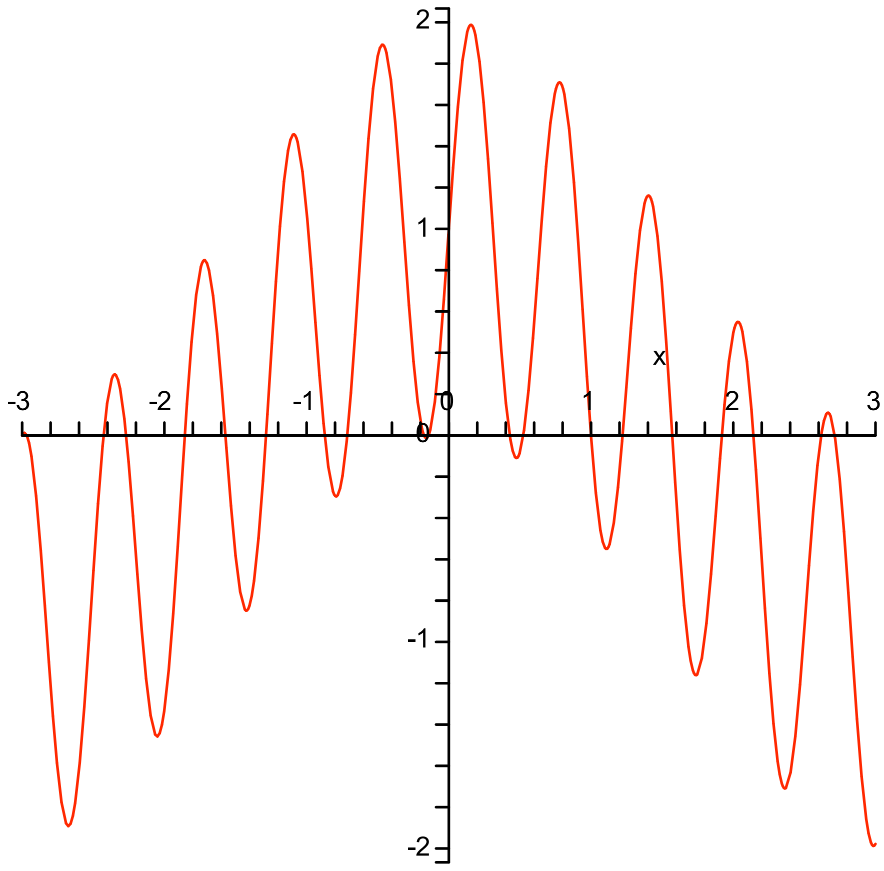
RootFinding

The new RootFinding package extends the functionality of the fsolve routine, particularly with regard to finding many or all roots of a system simultaneously.

The following is similar to an example posted to USENET within the last six months. Find all the real roots of the following: sin(10*x)+cos(x) over x in [-10..10] .

```
> restart;
  func := sin(10*x) + cos(x);
  plot(func,x=-3..3);
```

$$func := \sin(10\,x) + \cos(x)$$



```
> with(RootFinding):
  sol := Analytic(func,-3-0.001*I..3+0.001);
```

Have all the real roots been found?

```
sol := -0.14279966607226, -0.17453292519943, 0.42839899821679, 1.2217304763960,
       -1.2851969946504, 1.5707963267949, -1.8563956589394, 1.9198621771938,
       -2.2689280275926, 2.7131936553730, -2.9670597283904, -1.5707963267949,
       -0.87266462599716
```

```
> nops([sol]);
                                        13
```

```
>
```

An an exercise, use the fsolve command, with its avoid option, to repeatedly generate solutions containing more and more

distinct real roots in this range. Can it find all the roots? Is it faster, for finding all the roots? Is faster, for finding just a single

root?

References and links

Efficient LinearAlgebra Computations

RootFinding

Optimization

LinearAlgebra

Maplets

A Customizable Interface to Maple

Stephen Forrest, Maplesoft

sforrest@maplesoft.com

First Steps

Introduction to Maplets

What is a Maplet?

A Maplet consists of one or more windows which interact with the user by means of buttons, checkboxes, text fields, and other standard graphical controls.

Why bother with Maplets?

Maple is a powerful problem-solving environment capable of doing quite advanced mathematics, but the price of this power is considerable complexity.

With a Maplet, this complexity can be retained but restrained: a simple menu interface, tailored to a specific task or problem.

Examples

With these built-in Maple commands which launch Maplets, we will see a bit of what Maplets are capable of.

Computing Matrix Inverses

The following Maplet is designed for students learning to compute matrix inverses using Gaussian elimination; the point-and-click interface avoids much of the confusion over syntax beginning users face.

```
> Student:-LinearAlgebra:-InverseTutor():
>
```

This is not to say, however, that Maplets are designed to be used only by beginners. Building an interface customized for a specific task offers convenience and benefit even to those users who would otherwise have had no difficulty in performing the task in Maple.

Solving Differential Equations

In the following example we illustrate the use of a Maplet in solving a differential equation.

```
> eqn := diff(y(x),x,x)-10*(1-y(x)^2)*diff(y(x),x)+y(x) = 0;
> dsolve[interactive]( { eqn, y(0) = 2, D(y)(0) = 0 } ) ;
>
```

Building Plots

In this example we show how a Maplet may be used for plotting a function.

```
> plots[interactive]( x^2-y^2 );
>
```

First Maplet

```
> restart;
```

In this first example, we see a Maplet with one window. This window contains the text My First Maplet in the title bar, and in homage to the grand old tradition of programming tutorials, the body of the window has the text Hello, World!.

```
> use Maplets:-Elements in
      maplet := Maplet( Window( "My First Maplet", [
          ["Hello, world!"]
      ] ) );
  end use:
> Maplets:-Display( maplet );
>
```

First, notice the use Maplets:-Elements in part. The Maple identifiers used in building a Maplet are members of a Maple package named Maplets:-Elements , which is itself a subpackage of the package Maplets . We could have written the above code using full names (e.g. Maplets:-Elements:-Window) but for convenience we've enclosed the code in a use block. We'll see this throughout the examples.

Notice also that there are two separate Maple commands here: building a Maplet with a call to the Maplet command and assigning it to the name maplet, and actually running it with the Maplets:-Display command.

A Simple Layout

The visual design of the body of the window is created using a list of lists. Each list forms a separate row in the resulting window body. It is up to the Maplet design engine to place these objects in what is hopefully a visually pleasing arrangement.

```
> use Maplets:-Elements in
      maplet := Maplet( Window( "My Second Maplet", [
          ["Hello There"],
          ["1", "2", "3"],
          ["ABCDEFG", "HIJKLMNOPQR"],
          [Button( "OK", Shutdown() )]
      ] ) );
  end use:
```

```
  Maplets:-Display( maplet );
>
```

It is possible to exert finer control over the placement of objects in a window, but for most Maplets, the default is more than sufficient. We'll see later how we can make this layout more interesting.

The first non-text object in the above maplet is the button. That button has the text OK and when that button is clicked, the Shutdown action occurs.

There are many other such visual elements which can be placed into a Maplet window.

More Interesting Elements

In the following Maplet we see a number of the more interesting elements available for use.

```
> use Maplets:-Elements in
      maplet := Maplet( Window( "My 3rd Maplet", [
          [Button( "OK", Shutdown() ),
           CheckBox( "Some text" ),
           ComboBox( "Combo", ["Hello", "Goodbye"] ),
           DropDownBox( "DropDown",
              ["DropDown", "Hello", "Goodbye"]
           )
          ],
          ["ABCD", Label( "ABCD" )],
          [ListBox( ["First", "Second", "Third", "4th"] )],
          [MathMLViewer( sin(x)^2 + cos(x)^2 )],
          [Plotter( plot( sin(x), x=0..10 ) )],
          [Slider( 10..20, 15, showticks = true, minorticks = 1 ),
           TextBox(),
           TextField(),
           ToggleButton("Press Here")]
      ] ) );
  end use:
  Maplets:-Display( maplet );
>
```

Element Options

We've seen that we can choose from a wide variety of visual elements to include in a Maplet window. It's natural to ask how we can customize these elements to suit our particular needs.

Each Maplet element has a number of options which affect its behaviour and appearance. In looking at the previous Maplet, you might have noticed that one element was specified with an argument `showticks = true`. Other options modifying Maplet elements can be specified in similar ways.

```
> ?Button
```

```
> use Maplets:-Elements in
      maplet := Maplet(
          Window( "A Colourful Button",
                [[Button( "OK", background = blue, enabled = false,
                     tooltip = "Can't click me", Shutdown() )]]
          )
      );
  end use:
  Maplets:-Display( maplet );
```

```
>
```

A note on style: in general, one should be reasonable about customizing options. As much as you might love to colour an important button red, you probably shouldn't: users won't expect it, and they'll be uncomfortable if they see it.

References and State

Now, we can do much more than simply customizing the display of option. A number of Maplet elements are capable of holding state, such as the TextField in the following Maplet, which a user can modify:

```
> use Maplets:-Elements in
      maplet := Maplet( Window( "Text Maplet", [
          [TextField()],
          [Button( "OK", Shutdown() )]
      ] ) );
  end use:
  Maplets:-Display( maplet );
```

```
>
```

A more complete list is:

CheckBox
ComboBox
ListBox
MathMLEditor
Slider
TextBox
TextField
ToggleButton

The state information associated with a Maplet element is not restricted to things which a user can directory modify. For example, an element 'knows' whether or not it is visible.

We might like to programmatically ask the Maplet about one of its elements. In order to refer to such elements in order to ask such questions, we must identify them. We do this using references. In the previous example, we could replace the code `TextField()` swith `TextField["input"]`, which allows this element to be referenced later by the name "input" .

The reference may be a Maple string or symbol. Every element can be given a reference by indexing the element with the desired name or string. In general, it is safer to work with strings, but like everything else in Maple, symbols too are allowed.

The first use of a reference will be in the Shutdown action (of which we will see more later on):

```
> use Maplets:-Elements in
      maplet := Maplet( Window( "Text Maplet", [
          [TextField["input"]()],
          [Button( "OK", Shutdown( ["input"] ) )]
      ] ) );
  end use:
  Maplets:-Display( maplet );
>
```

Here, when the user clicks on OK, then the Shutdown action is triggered and returns a list. The one object in this returned list is the contents, or value, of the text field box referenced by **input**.

```
> use Maplets:-Elements in
      maplet := Maplet( Window( "Text Maplet", [
          [TextField[input]()],
          [TextField[input2]()],
          [Button( "OK",
             Shutdown( [input, input2, input(visible)] ) )
          ]
      ] ) );
  end use:
  Maplets:-Display( maplet );
>
```

What is 'visible'? Most elements have a default value option but each maplet has many other options. See TextField for a list of the options accepted by TextField.

From Maple code, you can access any of these options by using the convention **some_reference(option_name)**. If you do not include option_name with parentheses after specifying **some_reference**, and the element has a option named **value**, then that option is used.

Exercise 1

Part 1

Write a maplet which queries the user for his or her name, then returns it to the worksheet.

Part 2

Take this maplet and wrap it in a procedure which takes the returned name and prints Your name starts with X, where X is the first letter in the name which was returned.

Actions and Commands (Event Driven Programming)

With Maplets, you can define a default set of events (called actions) which occur when a maplet starts up. After that, nothing happens unless the user initiates an action by either clicking a button, moving a slider, or entering text in a field.

Most of the names of these action elements are rather self explanatory:

CloseWindow
Evaluate
RunDialog
RunWindow
SetOption
Shutdown

Shutdown exits a maplet, while **CloseWindow** only closes a maplet. By default, when the last window of a maplet is closed, the maplet exits and control is returned to the worksheet.

```
> use Maplets:-Elements in
      maplet := Maplet( Window( "My 4th Maplet", [
          ["Hello World"],
          [Button( "OK", Shutdown() )]
      ] ) );
  end use:
  Maplets:-Display( maplet );
```

```
>
```

We've seen before that we can give the Shutdown element a list of references to return. In addition to a list, it accepts a string as well; this is returned to the worksheet.

```
> use Maplets:-Elements in
      maplet := Maplet( Window( "My 5th Maplet", [
          ["Hello World"],
          [Button( "OK 1", Shutdown("Button 1") ),
           Button( "OK 2", Shutdown("Button 2") )]
      ] ) );
  end use:
  Maplets:-Display( maplet );
```

```
>
```

If you are returning both a value and a list of references, you will get an expression sequence with the first element being the return value and the second being a list of the references.

The next, and most important, command element is Evaluate. This calls a Maple procedure and allows the maplet to interact with the Maple kernel and library.

```
> Sure := proc()
     print( "Sure was pressed" );
  end proc:

  Maybe := proc()
     print( "Maybe was pressed" );
  end proc:

  use Maplets:-Elements in
      maplet := Maplet( Window( "My 5th Maplet", [
          [Button( "Sure", Evaluate( function = 'Sure()' ) ),
           Button( "Maybe", Evaluate( function = 'Maybe()' ) )
```

```
            ],
            [Button( "Exit", Shutdown() )]
        ] ) );
  end use:
  Maplets:-Display( maplet );
```

```
>
```

The SetOption element can be used to directly set an option of any maplet element within the maplet:

```
> use Maplets:-Elements in
      maplet := Maplet( Window( "Set Option", [
          [Button( "Enable",  SetOption( exit(enabled) = true ) ),
           Button( "Disable", SetOption( exit(enabled) = false ) ),
           Button[exit]( "Exit", Shutdown() )]
      ] ) );
  end use:
  Maplets:-Display( maplet );
```

```
>
```

You can use SetOption to clear a text field quite easily:

```
> use Maplets:-Elements in
      maplet := Maplet( Window( "Set Option", [
          [TextField[TF](),
           Button( "Clear", SetOption( TF = "" ) ),
           Button[exit]( "Exit", Shutdown() )]
      ] ) );
  end use:
  Maplets:-Display( maplet );
```

```
>
```

Communication between Maplets and the Maple Kernel

Once you call a Maple function, you can both query and update anything in a maplet. You do this by using the two tools Maplets:-Tools:-Get and Maplets:-Tools:-Set .

```
> use Maplets:-Elements in
      maplet := Maplet( Window( "Integrator", [
          ["Integrand: ", TextField[integrand]()],
          ["Integral:  ", TextField[integral]( editable = false )],
          [Button( "Integrate",
                Evaluate( function = 'Integrate()' )
           ),
           Button( "OK", Shutdown([integral]) )]
      ] ) );
  end use:
```

```
> Integrate := proc()
      local expr;
```

```
        expr := Maplets:-Tools:-Get( integrand::algebraic );
        expr := int( expr, x );
        Maplets:-Tools:-Set( integral::algebraic = expr );
    end proc:
    Maplets:-Display( maplet );
> 
```

Error Handling

Let's take a look at that last example. What happens if the user does something wrong?

The most important tool we have for this purpose is the try-catch construct:

```
> Integrate := proc()
        local expr;
        try
            expr := Maplets:-Tools:-Get( integrand::algebraic );
        catch:
            return; # message?
        end try;

        try
            expr := int( expr, x );
        catch:
            return; # message?
        end try;

        Maplets:-Tools:-Set( integral::algebraic = expr );
    end proc:
> Maplets:-Display( maplet );
> 
```

Exercise 2

Part 1

Ask the user to enter an integer n in a TextBox. Let the user test whether n is prime by clicking on a button, and update another TextBox with this information.

Part 2

To the previous Maplet, add a DropDownBox with which the user can choose his or her favourite prime factor of the number n. Make this dropdown box inactive when the Maplet starts, and activate it and fill its contents when the user tests the primality of n.

Extra: Activate the dropdown box only if n has more than one prime factor.

The Finer Details

More than One Window

Up until now, we've seen only one window. A Maplet is not just a window: a window is merely one component of a Maplet, and so therefore it is reasonable that a Maplet has more than one window.

Now, if you only include one window, the Maplet assumes you want to run that window on starting. If you have more than one, it's not sure, and returns an error message:

```
> use Maplets:-Elements in
      maplet := Maplet(
         Window( "Hi",  [["Hi"]] ),
         Window( "Bye", [["Bye"]] )
      );
  end use:
```

This introduces us to more complex actions. In the following Maplet, we specify which window we'd like to start with:

```
> use Maplets:-Elements in
      maplet := Maplet( onstartup = RunWindow( W1 ),
        Window[W1]( "Hi", [["Hi"]] ),
        Window[W2]( "Bye", [["Bye"]] )
      );
  end use:
  Maplets:-Display( maplet );
```

```
>
```

Below is a slightly more complicated example. Here we have two commands running as a result of one action: clicking the button.

```
> use Maplets:-Elements in
      maplet := Maplet( onstartup = RunWindow( W1 ),
          Window[W1]( "Hi", [[
              Button( "OK",
                  Action( CloseWindow( W1 ), RunWindow( W2 ))
              )
          ]] ),
          Window[W2]( "Bye", [[Button( "Okay", Shutdown() )]] )
      );
  end use:
  Maplets:-Display( maplet );
```

```
>
```

Dialogs

Maplets come with a number of preset dialog elements.

```
> ?Maplets,DialogElements
```

```
> use Maplets:-Elements in
      maplet := Maplet( AlertDialog(
          "Assuming x > 0 leads to a contradiction",
          'onapprove' = Shutdown("true"),
          'oncancel' = Shutdown("FAIL")
      ) );
  end use:
  Maplets[Display](maplet);
>
```

One thing to note: dialog elements do not have a state, so you cannot query the value that it returns.

```
> use Maplets:-Elements in
      maplet := Maplet( ColorDialog[D1](
          'onapprove' = Shutdown( [D1] ),
          'oncancel' = Shutdown()
      ) );
  end use:
  Maplets[Display]( maplet );
>
```

Menus

Menus are just organized buttons. It is recommended that you follow standard conventions if you decide to include menus in your maplet.

```
> use Maplets:-Elements in
      maplet := Maplet( onstartup = RunWindow( W1 ),
           Window[W1]( "With Menu",
           MenuBar( Menu( "File",
               MenuItem( "Hello", RunDialog( D1 ) ),
               MenuItem( "Exit", Shutdown() )
           ) ),
           [["Some Text"]]
      ), ConfirmDialog[D1]( "Are you fine?" ) );
  end use:
  Maplets:-Display( maplet );
>
```

You can use the menu separator to place a line between adjacent menu items. Use these to group common menu items and don't over do it...

Toolbars

Toolbars are like menus, except that they have buttons, as you can see in the worksheet environment.

```
> use Maplets:-Elements in
     maplet := Maplet(
        Window('title' = "Integration w.r.t. x",
           'toolbar' = ToolBar(
               ToolBarButton("Exit", Shutdown()),
                ToolBarSeparator(),
                ToolBarButton("Do It",
                    'onclick'=Evaluate('TF1' = 'int(TF1, x)'))
            ),
            [TextField['TF1'](),
             Button("OK", Shutdown("OK"))]
         )
      ):
  end use:
  Maplets[Display](maplet):
>
```

Pop-up Menus

The text field and text box elements takes a popupmenu option, which specifies a PopupMenu element.

```
> use Maplets:-Elements in
      maplet := Maplet( Window( "Integrator", [
          ["Integrand: ", TextField[integrand](
              popupmenu = PopupMenu(
                  MenuItem( "Integrate",
                      Evaluate( function = 'Integrate()' ) )
              )
          )],
          ["Integral:  ", TextField[integral]( editable = false )],
          [Button( "OK", Shutdown([integral]) )]
      ] ) );
  end use:
```

```
> Integrate := proc()
      local expr;
      expr := Maplets:-Tools:-Get( integrand::algebraic );
      expr := int( expr, x );
      Maplets:-Tools:-Set( integral::algebraic = expr );
  end proc:
```

```
> Maplets:-Display( maplet );
```

```
>
```

Fancier Layouts

Okay, what I said before about formatting windows wasn't the whole truth.

We don't have to have a list of lists. Instead, what Maple does is interpret the first list as a list of elements to be displayed vertically. If that list contains any sublists, those are to be displayed horizontally, and so on, always alternating between vertical and horizontal display:

```
> use Maplets:-Elements in
      maplet := Maplet( Window( "Fancy", [
          "ABC",
          ["DEF", "GHI", ["J", "K", "L"]],
          ["M", ["OP", "QR", ["S", "T", "U"]]]
      ] ) );
  end use:
```

```
> Maplets:-Display( maplet );
```

Borders and Captions

We can add borders and captions to the various rows and columns of maplet elements.

```
> use Maplets:-Elements in
      maplet := Maplet( Window( "Fancy", [
          "ABC",
          ["DEF", "GHI", BoxRow( border = true, "J", "K", "L" )],
          ["M", BoxRow( border = true, caption = "Hello",
                        "OP", "QR", ["S", "T", "U"] )]
      ] ) );
  end use:
```

```
> Maplets:-Display( maplet );
```

```
> ?Maplets,LayoutElements
```

```
> plots[interactive]( sin(x) );
```

Elements in Detail

```
> ?Maplets,WindowElements
```

```
>
```

More Exercises

The following are suggested exercises to help you learn about Maplets. Choose any of the following exercises that interest you, or write your own Maplet that does something else entirely.

Exercise 3

Write a procedure which, takes two subsets of {a, b, c, d, e} and asks the user if the first is a subset of the second. Allow the user to click Yes or No, and open a new Maplet which indicates whether the user was right or wrong.

If the user was wrong, optionally try to discuss why he or she was wrong. E.g., {a, b, c} is not a subset of {a, b, d} because c is not an element of {a, b, d}.

Exercise 4

Pick a Maple function and write a Maplet interface to that function. See Maplets[Examples] for some ideas on design of your Maplet.

Exercise 5

As the user to integrate a random polynomial (you can generate one by using the randpoly command).

Check the user's answer using whatever Maple tool you find appropriate. Tell the user if he or she is correct.

Exercise 6

Play the high-low game with the user. Select a random number from 0 to 100, and ask the user to guess the value. Tell the user how far off he or she was from the optimal number of guesses.

Exercise 7

Use the slider zoom in on a plot.