Algorithmic Challenges in Polynomial and Linear Algebra
Défis algorithmiques dans l'algèbre polynomiale et l'algèbre linéaire
(Org: **Stephen Watt** (Western))

CARLOS BELTRAN, University of Toronto, Department of Mathematics, Toronto, Ontario M5S 2E4, Canada
On the complexity of approximating solutions of systems of polynomial equations

Let $f = (f_1, \ldots, f_n)$ be a system with n equations and n complex unknowns. An approximate zero of f is an affine point such that the successive iterations of Newton's method converge quadratically to an exact zero of the system.

I will present an Average Las Vegas numerical procedure, joint work with L. M. Pardo, guaranteed to produce approximate zeros of systems of equations in polynomial time, on the average.

I will also introduce some very recent results, joint work with M. Shub, that suggest the existence of much faster algorithms, with average running time almost linear in the size of the input. The search of such an algorithm is linked to the study of a geometrical problem: The geodesics in the "condition number metric". Open questions, both from complexity and geometrical perspectives, will be discussed.

JON BORWEIN, Dalhousie Faculty of CS
Computationally discovered and proved generating functions

This lecture will describe older and very recent work [2], [4] in which Bailey, Bradley and I hunted for various desired generating functions for zeta functions and then were able to methodically prove our results.

One example is

$$3\sum_{k=1}^{\infty} \frac{1}{\binom{2k}{k}(k^2-x^2)} \prod_{n=1}^{k-1} \frac{4x^2-n^2}{x^2-n^2} = \sum_{n=1}^{\infty} \frac{1}{n^2-x^2} \tag{1}$$
$$\left[= \sum_{k=0}^{\infty} \zeta(2k+2)\, x^{2\,k} \; = \frac{1-\pi x \cot(\pi x)}{2x^2} \right].$$

The constant term in (1) recovers the well known identity

$$3\sum_{k=1}^{\infty} \frac{1}{\binom{2k}{k}k^2} = \sum_{n=1}^{\infty} \frac{1}{n^2} = \zeta(2).$$

Equivalently, for each positive integer k one has the *generalized hypergeometric* identity

$${}_3\mathrm{F}_2\left(\left.\begin{matrix}3k, -k, k+1 \\ 2k+1, k+\frac{1}{2}\end{matrix}\right| \frac{1}{4}\right) = \frac{\binom{2k}{k}}{\binom{3k}{k}}. \tag{2}$$

As I hope to show, discovering (1) and then proving (2) formed one of the most satisfying experimental mathematics experiences I have had. I will also describe more recent work to appear in [3, 2008] regarding

$${}_3\mathrm{F}_2\left(\left.\begin{matrix}3k, -k, k+1 \\ 2k+1, k+\frac{1}{2}\end{matrix}\right| 1\right). \tag{3}$$

References

[1] D. H. Bailey and J. M. Borwein, *Experimental Mathematics: Examples, Methods and Implications.* Notices Amer. Math. Soc. **52**(2005), 502–514.

[2] David Bailey, Jonathan Borwein and David Bradley, *Experimental Determination of Apéry-type Formulae for* $\zeta(2n+2)$. Experiment. Math. **15**(2006), 281–289. [D-drive Preprint 295]

[3] Jonathan M. Borwein and David H. Bailey, *Mathematics by Experiment: Plausible Reasoning in the 21st Century.* A. K. Peters, Natick, MA, 2004; second extended edition, 2008.

[4] Jonathan M. Borwein, David H. Bailey and Roland Girgensohn, *Experimentation in Mathematics: Computational Paths to Discovery.* A. K. Peters, Natick, MA, 2004.

JACQUES CARETTE, McMaster University, Department of Computing and Software

Algorithm families, or how to write less code that does more

Program families are well-known in software engineering, and generic algorithms are pervasive in computer algebra. Nevertheless, in computer algebra libraries, one finds an inordinate amount of code duplication, even at the level of named algorithms. Some techniques are already in use to deal with this issue, with C++ template programming in common use in Linbox and post-facto extensions in Aldor.

By combining ideas from program families and generic algorithms, a careful study of the variations of well-known algorithms allows us to re-unite these offshoots back into a coherent family, where all members find their place. This talk will mostly concentrate on showing how many dimensions exist in simple families of algorithms, with LU decomposition being the guiding example. While previous work has clearly documented between 4 and 6 dimensions, we have discovered at least 19 separate "dimensions of variation" in that same family. As time permits, we will show that our chosen implementation method (higher-order modules of code generators in MetaOCaml) possesses some technical advantages over other solutions.

This is joint work with Oleg Kiselyov.

WAYNE EBERLY, Department of Computer Science, University of Calgary

On the Reliability of Block Wiedemann and Lanczos Algorithms—Another Piece of the Puzzle

Block Wiedemann and Lanczos algorithms are now available as part of the Linbox package. These have been proved to be reliable in limited cases, or more generally if matrix preconditioners are used to avoid pathological inputs. On the other hand, considerably simpler heuristics (based on the same ideas) have been used with considerable success for various number-theoretic computation. This motivates work to generalize the proofs of reliability that we now have by weakening or eliminating assumptions about the input upon which these proofs currently depend.

Various claims about the reliability of such algorithms depend on bounds for the expected nullity of various "block Hankel" matrices. These are matrices that are generated using a matrix $A \in \mathsf{F}^{N\times N}$ with entries in a field F — generally part of the input for the given problem — as well as blocks of vectors $u, v \in \mathsf{F}^{N\times k}$ that are randomly generated. Given A, u, and v, the (i,j)-th block of the matrix that is of interest is the $k \times k$ matrix $u^T A^{i+j} v$.

Bounds on the nullity of such matrices are now available in a reasonably general case, namely, that the number of nontrivial invariant factors of the matrix A is less than the "blocking factor" k. In this talk I will sketch a proof that this assumption about invariant factors is not, in fact, necessary: Useful bounds on the nullity of the above block Hankel matrices can be obtained for arbitrary matrices $A \in \mathsf{F}^{N\times N}$.

WILLIAM M. FARMER, McMaster University, Hamilton, ON
Formalizing the Context in Computational Mathematics

The set of vocabulary, background assumptions, and rules that govern an application of mathematics is called its *context*. The context of an application in deductive mathematics is traditionally formalized as an *axiomatic theory* in which mathematical knowledge is represented declaratively as a set of *axioms*. The context of an application in computational mathematics is often formalized as an *algorithmic theory* in which mathematical knowledge is represented procedurally as a set of *algorithms*. The background assumptions of an algorithmic theory and the specifications of the algorithms are usually not an explicit part of an algorithmic theory.

As a result, an algorithmic theory can be used to perform computations, but it cannot be used to understand what the results of the computations mean. A *biform theory* can represent mathematical knowledge both declaratively and procedurally. In particular, it can include algorithms equipped with precise specifications of their input-output relationships. In this talk, we will explain what a biform theory is and how a biform theory can be used to formalized the context in computational mathematics. We will also briefly introduce Chiron, a general-purpose logic based on set theory that is especially well-suited for expressing biform theories.

MARK GIESBRECHT, University of Waterloo, Waterloo, Ontario, Canada
New Algorithms for Lacunary Polynomials

Some of the most compelling theoretical work in computer algebra of the past decade has been on computations with lacunary or super-sparse polynomials. Computer algebra systems (and mathematicians!) are good at representing multivariate polynomials of very high degree but only a few terms in the form of a linked list of coefficient/exponent tuples. However, the repertoire of algorithms for computing with polynomials in this representation is limited, and the area is fraught with intractable problems. Recent work of Lenstra, Kaltofen and Koiran has extended the reach of fast algorithms. We seek here to further augment the available toolkit.

We give new algorithms for two important problems. First, we give an algorithm to interpolate an unknown integer polynomial in the sparsest shifted power basis. We assume that we are given a function or "black box" for evaluating that polynomial at a point modulo a prime or at a (complex) root of unity. Second, the question of functionally decomposing a univariate polynomial is investigated when the input is lacunary. A new algorithm is proposed to compute sparse decompositions. Some interesting connections are noted to long-standing open problems studied by Erdős, Schinzel and others.

This is work with Daniel Roche (Waterloo).

MICHAEL JACOBSON, University of Calgary
Computing the Regulator of a Real Quadratic Field

The regulator is an important invariant of a real quadratic number field, due to its close connection to fundamental solutions of the Pell equation and to cryptographic applications using such fields. We will describe state-of-the-art methods for computing the regulator that are effective for fields with discriminants as large as 100 decimal digits. These methods use the index-calculus strategy, and as such require time-consuming linear algebra calculations over the integers, which will be highlighted in this talk.

ILIAS KOTSIREAS, Wilfrid Laurier University, 75 University Avenue West, Waterloo, Ontario N2L 3C5, Canada
Systems of Polynomial Equations in Combinatorial Design Theory

Combinatorial Design Theory is a rich source of challenging systems of polynomial equations, mainly of dimension zero. We will describe some classes of such systems as well as some of the different methods that are used to tackle them. These methods

come from different disciplines, such as Computer Algebra, Artificial Intelligence, Combinatorics and Discrete Mathematics, while Supercomputing is often a useful aid.

GEORGE LABAHN, Cheriton School of Computer Science
Conditioning of the Generalized Hankel Eigenvalue Problem

Problems such as the construction of a sparse black box polynomial and finding poles of Padé approximants are examples of a generalized eigenvalue problem for Hankel matrices. In this talk we discuss the the numerical issues that come up when such problems are given in fixed precision arithmetic. A basic tool in this study is information on the condition number of Vandermonde matrices with complex entries.

This is joint work with B. Beckermann and G. Golub.

SONGXIN LIANG, University of Western Ontario
A New Maple Package for Solving Parametric Polynomial Systems

A new Maple package `RootFinding[Parametric]` for solving parametric systems of polynomial equations and inequalities is described. The main idea for solving such a system is as follows. The parameter space $\mathbb{R}^d$ is divided into two parts: the discriminant variety W and its complement $\mathbb{R}^d \backslash W$. The discriminant variety is a generalization of the well-known discriminant of a univariate polynomial and contains all those parameter values leading to non-generic solutions of the system. The complement $\mathbb{R}^d \backslash W$ can be expressed as a finite union of open cells such that the number of real solutions of the input system is constant on each cell. In this way, all parameter values leading to generic solutions of the system can be described systematically. The underlying techniques used are Gröbner bases, polynomial real root finding, and Cylindrical Algebraic Decomposition. This package offers a friendly interface for scientists and engineers to solve parametric problems, as illustrated by an example from control theory.

JOHN MAY, Maplesoft, 615 Kumpf Drive, Waterloo, Ontario
A Symbolic-Numeric Approach to Computing Inertia of Products of Matrices of Rational Numbers

Consider a rational matrix, particularly one whose entries have large numerators and denominators but which is presented as a product of very sparse matrices with relatively small entries. We will discuss symbolic-numeric hybrid strategies to compute the inertia (the number of positive, negative, and zero eigenvalues) of such a matrix in the nonsingular case that effectively exploits the product structure. The method works by computing the numerical LU decomposition of the product with a succession of floating point QR decompositions of the small, sparse, factors in the product representation.

This is joint work with B. David Saunders, and David H. Wood in the Department of Computer Science at the University of Delaware.

MARC MORENO-MAZA, University of Western Ontario, London, Canada
Triangular decomposition of polynomial systems: from practice to high performance

Triangular decompositions are one of the most studied techniques for solving polynomial systems symbolically. Invented by J. F. Ritt in the early '30s for systems of differential polynomials, their stride started in the late '80s with the method of W. T. Wu dedicated to algebraic systems. Different concepts and algorithms extended the work of Wu. At the end of the '90s the notion of a regular chain, introduced by M. Kalkbrener, led to important algorithmic improvements. The era of polynomial system solvers based on triangular decompositions could commence.

Since 2000, several promising complexity results and algorithmic discoveries have stimulated the development of implementation techniques. These have permitted fast polynomial arithmetic and parallelism to speed up key subroutines substantially. Today,

new algorithms for computing triangular decompositions are being invented to make the best use of these techniques. I anticipate that triangular decomposition solvers will become an invaluable and irreplaceable support to scientific computing.

ERIC SCHOST, University of Western Ontario, London, ON
Conversion algorithms for orthogonal polynomials

Families of orthogonal polynomials satisfy recurrence relations which are often the key to the design of fast algorithms. In this talk, we consider conversion problems, from an orthogonal basis to the monomial basis and conversely. Using the recurrence relation on orthogonal polynomials, one easily comes up with efficient algorithms for the direct conversion. We propose a fast algorithm for the converse direction, exploiting classical continued fractions identities and algorithm transposition techniques to reach a quasi-linear complexity.

Joint work with Bruno Salvy and Alin Bostan.

ARNE STORJOHANN, University of Waterloo
Faster algorithms for the Frobenius canonical form

Like the better known Jordan form, the Frobenius form of a square matrix A over a field K is a unique representative of the set of all matrices similar to A. The Frobenius form captures completely the geometric structure of a matrix, and clearly reveals invariants such as the minimal and characteristic polynomial.

The problem of computing the form has been very well studied. Let $2 < \theta \leq 3$ be such that $O(n^\theta)$ operations in K are sufficient to multiply together two matrices in $\mathsf{K}^{n\times n}$. Over a sufficiently large field, with $\#\mathsf{K} \geq n^2$, Giesbrecht's (1993) randomized algorithm computes the Frobenius form F, together with a similarity transformation matrix U such that $F = U^{-1}AU$, using an expected number of $O(n^\theta \log n)$ operations in K. More recently, Eberly (2000) describes a randomized algorithm, applicable over a field K of any size, that computes both the form and a transformation matrix using an expected number of $O(n^\theta \log n)$ operations in K.

In this talk I describe a new randomized algorithm for computing the Frobenius form. Over a sufficiently large field the new algorithm uses an expected number of $O(n^\theta)$ field operations, thus improving by a factor of $\log n$ on previously known results. Once the Frobenius form itself has been computed, a similarity transformation matrix can be constructed using an additional $O(n^\theta \log\log n)$ field operations.

Joint work with Clément Pernet.

YUZHEN XIE, The University of Western Ontario
Solving Polynomial Systems Symbolically and in Parallel

We present algorithms and an implementation framework for solving polynomial systems symbolically and in parallel.

Symbolic methods are powerful tools in scientific computing. The implementation of symbolic solvers is, however, highly difficult. Indeed, they are extremely space and time consuming when applied to large examples. The increasing availability of parallel and distributed architectures offers the opportunity to realize high-performance solvers that can largely expand the range of solvable problems.

We have developed a high-level categorical parallel framework, written in the Aldor language, to support high-performance computer algebra on symmetric multi-processor machines and multicores. A component-level parallel solver for computing triangular decompositions has been realized using this framework. Our solutions address the following challenges:

(1) creation of coarse-grained level parallelism aiming to achieve multi-level parallelization for symbolic solving over clusters of multiprocessor machines;

(2) efficient management and scheduling of highly dynamic and irregular tasks; and

(3) effective data-communication of sophisticated mathematical objects.

Our work in progress focuses on the study of efficient memory usage in parallel symbolic solving. Memory consumption appears to be one of the bottlenecks in our parallel solver: the concurrent threads or processes incur more space usage than a sequential execution due to distribution and/or duplication of mathematical objects. We discuss approaches to face this new challenge.