Symbolic Computation (Computer Algebra)

James Davenport

University of Bath

16 November 2015

Preamble

Generally talking about "polynomial" computer algebra:
Major packages: Maple and Mathematica (commercial); Reduce, Macsyma; meta-package Sage
There's also the group theory end of the world: GAP and MAGMA
Many specialist packages: F5, Singular and CoCoA for Gröbner bases, QEPCAD, RedLog for real solving, and I've doubtless omitted many others

Notation (for complexity purposes)

- m number of polynomials
- n number of variables
- d maximum degreee (in each variable separately)
- l bit-length of coefficients (we will often not bother with this, as most solving algorithms are $O(l^3)$)
- t Number of non-zero terms.

(Dense) polynomials

"Obviously" a vector of coefficients

Addition $O(dl)$

Multiplication School: $O(d^2 l^2)$; Karatsuba: $O((dl)^{1.585})$; FFT: $O(dl \log(dl) \log\log l)$

Division the same

GCD the same $\times \log d$

But most problems aren't dense: a dense polynomial has $(d+1)^n$ terms.

(Sparse) polynomials [DC10]

"Obviously" a list of (exponent,coefficient) pairs

Addition $O(tl)$

Multiplication School: $O(t^3 + t^2 l^2)$; Better $O(t^2(l^2 + \log t))$

Division not the same at all: $\frac{x^n-1}{x-1} = x^{n-1} + \cdots + 1$

D with remainder School: $O(d^3 t l^2)$, better $O(d^2 t(l^2 + \log d))$

Exact D We can stop if the coefficients grow too big [ABD85] $O(dt(dl^2 + \log\min(d,t)))$; in fact $O(t_f + t_{f/g} t_g (l^2 + \log\min(t_{f/g} t_g))$ [MP11]

GCD $O(d^4 l^2)$: some dependence on d is inherent [Sch03]:

$$\gcd(x^{pq}-1, x^{p+q}-x^p-x^q+1) = x^{p+q-1}-x^{p+q-2}\pm\cdots-1$$

Open An algorithm for $\gcd(f,g)$ polynomial in $t_f, t_g, t_{\gcd(f,g)}$.

In practice

We compute polynomial gcd via modular methods

1. Compute $\gcd(f, g)$ modulo several p_i
   - ? How many? *A priori* bounds are usually too high, so the best algorithms are adaptive
2. Discard those that have too high a degree
3. Use Chinese Remainder Theorem to produce $\gcd(f, g) \pmod{\prod' p_i}$
4. Interpret over **Z and check**

Same technique works in several variables, using $y - v_i$ as "primes"
The real challenge is multivariate sparsity: [Zip79] has an algorithms that seems to be polynomial in d, t (and not d^n)

Polynomial Factoring (univariate)

- Can't use modular methods, as no idea which factor goes with which ($\mathbf{Z}[x]/\prod p_i$ is not a unique factorisation domain)
- There are irreducible polynomials (e.g. x^4+1) which factor compatibly modulo every prime (no good primes!)

! rare in theory but common in practice [ABD85], especially with algebraic numbers

- Trying every combination of mod p factors is exponential
- LLL was invented to make this polynomial, but $O(d^{12})$
- There are better lattice-based methods these days [vH02]

Polynomial Factoring (multivariate)

- For reductions from multivariate to bivariate

  (by replacing all the other variables by values)

  almost all evaluations are good, in that the factorisation is the same after evaluating
- In practice we see the same reducing to univariate

Hence the real challenge is univariate, in theory

In practice there are substantial challenges especially over sparsity [Wan78]

But Factoring is still best avoided, and implementations try to: replacing "irreducible" by "square-free and relatively prime" where possible

Real Roots

Univariate A polynomial has d roots

Computed Sturm sequences, Thom's lemma, Descartes Rule

Sparse A polynomial has $\leq 2t - 1$ real roots

(Almost) no good, i.e. $O(t^k)$ not $O(d^k)$, algorithms for isolating these

Multivariate again, should have low real complexity, but "change of coordinates" destroys sparsity.

Resultants and Discriminants

Resultant $R(\mathbf{x}) := \mathrm{Res}_y(f(\mathbf{x},y), g(\mathbf{x},y))$ has as roots those $\mathbf{x} : \exists y \in \mathbf{C} : f(\mathbf{x},y) = g(\mathbf{x},y) = 0$: common zeros = crossings of surfaces

Discriminant $D(\mathbf{x}) = \mathrm{Disc}_y(f(\mathbf{x},y)) = \frac{1}{lc_y(f)} \mathrm{Res}(f, \frac{\partial f}{\partial y})$ has as roots those $\mathbf{x} : \exists y \in \mathbf{C} : f(\mathbf{x},y)$ has a double root: self-crossings or doubling-back

Degrees $O(d^2)$ and classical computing time $O(d^9 l^2)$ — modular methods are generally used, but still expensive
$\mathrm{Res}(f_1 f_2, g) = \mathrm{Res}(f_1, g)\,\mathrm{Res}(f_2, g)$;
$\mathrm{Disc}(f_1 f_2) = \mathrm{Disc}(f_1)\,\mathrm{Disc}(f_2)\,\mathrm{Res}^2(f_1, f_2)$, hence we want to keep polynomials in factored form
Iterated resultants also tend to factor also [BM09], generally into a "good part" and a "bad part"
No good interaction with sparsity!

Many views and uses, but we can consider them as non-linear analogue of Gaussian reduction to upper triangular form, **but**:

- Might have more equations than variables: $\{x^2-1,(x-1)(y-1),y^2-1\}$ — Back-substitution by Gianni–Kalkbrener Theorem [Gia89, Kal89]
- This doesn't work in dimension >0 [FGT01]
- Can have "mixed dimension" varieties: $\langle (x+1-y)(x-6+y),(x+1-y)(y-3)\rangle$ has solution $x=y-1$ **and** $(x,y)=(3,3)$
- Theoretical complexity results are very bad, but these inputs are "rare"
- Modular (Chinese Remainder) approaches are difficult

History of Quantifier Elimination

- In 1930, Tarski discovered [Tar51] that the (semi-)algebraic theory of $\mathbf{R}^n$ admitted quantifier elimination

  $\exists x_{k+1} \forall x_{k+2} \ldots \Phi(x_1, \ldots, x_n) \equiv \Psi(x_1, \ldots, x_k)$
- "Semi" = "allowing $>$, $\leq$ and $\neq$ as well as $=$"
- Needed as $\exists y : x = y^2 \Leftrightarrow x \geq 0$
- The complexity of this was indescribable
- In the sense of not being elementary recursive!
- In 1973, Collins [Col75] discovered a much better way:
- Complexity (m polynomials, degree d, n variables, coefficient length l)

$$(2d)^{2^{2n+8}} m^{2^{n+6}} l^3 \tag{1}$$

- Construct a cylindrical algebraic decomposition of $\mathbf{R}^n$, sign invariant for every polynomial
- Then read off the answer

What is a CAD?

A Cylindrical Algebraic Decomposition (CAD) is a mathematical object. Defined by Collins who also gave the first algorithm to compute one. A CAD is:

- a decomposition meaning a partition of $\mathbf{R}^n$ into connected subsets called cells;
- (semi-)algebraic meaning that each cell can be defined by a sequence of polynomial equations and inequalities;
- cylindrical meaning the cells are arranged in a useful manner — their projections are either equal or disjoint.

In addition, there is (usually) a sample point in each cell, and an index locating it in the decomposition

“Read off the answer”

- Each cell is sign invariant, so the the truth of a formula throughout the cell is the truth at the sample point.
- $\forall x F(x) \Leftrightarrow$ “$F(x)$ is true at all sample points”
- $\exists x F(x) \Leftrightarrow$ “$F(x)$ is true at some sample point”
- $\forall x \exists y F(x, y) \Leftrightarrow$ “take a CAD of $\mathbf{R}^2$, cylindrical for y projected onto x-space, then check

$$\forall \text{ sample } x \ \exists \text{ sample } (x, y) : F(x, y) \text{ is true"}: \text{ finite check}$$

NB The order of the quantifiers defines the order of projection

So all we need is a CAD!

$$
\begin{tikzcd}
\mathbb{R}^n \arrow[r, dotted, no head] \arrow[d] & \mathbb{R}^n \\
\mathbb{R}^{n-1} \arrow[d] & \mathbb{R}^{n-1} \arrow[u] \\
\mathbb{R}^{n-2} \arrow[d, dashed] & \mathbb{R}^{n-2} \arrow[u] \\
\mathbb{R}^1 \arrow[r] & \mathbb{R}^1 \arrow[u, dashed]
\end{tikzcd}
$$

Projection

Lifting
(& Isolation)

Root Isolation

So how do we project? (Lifting is in fact relatively straight-forward)

Given polynomials $\mathcal{P}_n = \{p_i\}$ in $x_1, \ldots, x_n$, what should $\mathcal{P}_{n-1}$ be?

Naïve (Doesn't work!) Every $\mathrm{Disc}_{x_n}(p_i)$, every $\mathrm{Res}_{x_n}(p_i, p_j)$

i.e. where the polynomials fold, or cross: misses lots of "special" cases

[Col75] First enlarge $\mathcal{P}_n$ with all its reducta, then naïve plus the coefficients of $\mathcal{P}_n$ (with respect to x_n) the principal subresultant coefficients from the Disc_{x_n} and Res_{x_n} calculations

[Hon90] a tidied version of [Col75].

[McC88] Let $\mathcal{B}_n$ be a squarefree basis for the primitive parts of $\mathcal{P}_n$. Then $\mathcal{P}_{n-1}$ is the contents of $\mathcal{P}_n$, the coefficients of $\mathcal{B}_n$ and every $\mathrm{Disc}_{x_n}(b_i)$, $\mathrm{Res}_{x_n}(b_i, b_j)$ from $\mathcal{B}_n$

[Bro01] Naïve plus leading coefficients (not squarefree!)

Are these projections correct?

[Col75] Yes, and it's relatively straightforward to prove that, over a cell in $\mathbf{R}^{n-1}$ sign-invariant for $\mathcal{P}_{n-1}$, the polynomials of $\mathcal{P}_n$ do not cross, and define cells sign-invariant for the polynomials of $\mathcal{P}_n$

[McC88] 52 pages (based on [Zar75]) prove the equivalent statement, but for order-invariance, not sign-invariance, provided the polynomials are well-oriented, a test that has to be applied during lifting.

But if they're not known to be well-oriented?

[McC88] suggests adding all partial derivatives

In practice hope for well-oriented, and if it fails use Hong's projection.

[Bro01] Needs well-orientedness and additional checks

What about the complexity?

If the McCallum projection is well-oriented, the complexity is

$$(2d)^{n2^{n+7}} m^{2^{n+4}} l^3 \tag{2}$$

versus the original

$$(2d)^{2^{2n+8}} m^{2^{n+6}} l^3 \tag{1}$$

and in practice the gains in running time can be factors of a thousand, or, more often, the difference between feasibility and infeasibility

"Randomly", well-orientedness ought to occur with probability 1, but we have a family of "real-world" examples (simplification/ branch cuts) where it often fails

Need it be this hard?

The Heintz construction

$$\Phi_k(x_k, y_k) :=$$
$$\exists z_k \forall x_{k-1} y_{k-1} \left[\begin{array}{c} y_{k-1} = y_k \wedge x_{k-1} = z_k \vee y_{k-1} = z_k \wedge x_{k-1} = x_k \\ \Rightarrow \Phi_{k-1}(x_{k-1}, y_{k-1}) \end{array} \right]$$

If $\Phi_1 \equiv y_1 = f(x_1)$, then $\Phi_2 \equiv y_2 = f(f(x_2))$,
$\Phi_3 \equiv y_3 = f(f(f(f(x_3))))$
[DH88] shows $\Omega\left(2^{2^{(n-2)/5}}\right)$ (using $y_R + iy_I = (x_R + ix_I)^4$)
[BD07] shows $\Omega\left(2^{2^{(n-1)/3}}\right)$ (using a sawtooth)
Hence doubly exponential is inevitable, but there's a lot of room!
In fact, there are theoretical algorithms which are singly-exponential in n, but doubly-exponential in the number of $\exists\forall$ alternations

Useful special cases

[McC99] "equational constraints" : when $\Phi \equiv f(x, y, \ldots) = 0 \wedge (\ldots)$

Note If $\Phi \equiv (f_1(x, y) = 0 \wedge g_1(x, y) < 0) \vee (f_2(x, y) = 0 \wedge g_2(x, y) < 0$, which has no obvious equational constraint, we can consider $(f_1 \cdot f_2)(x, y) = 0 \wedge \Phi$, which is equivalent (but higher degree)

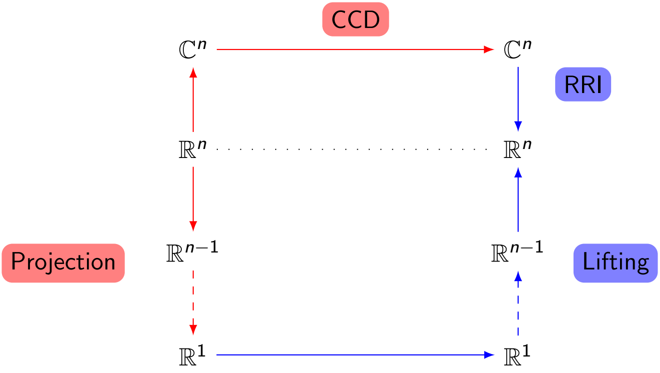
[BDE+13] "truth table invariant CAD" treats this directly

[BDE+14] also handles the case where not every clause has an equality (TTICAD)

Roughly speaking, the effect is to reduce n by 1, which square roots the complexity

An alternative approach [CMXY09]

Proceed via the complex numbers,



Do a complex cylindrical decomposition via Regular Chains
Can be combined with truth table ideas [EBC$^+$14]

Example Complex CD

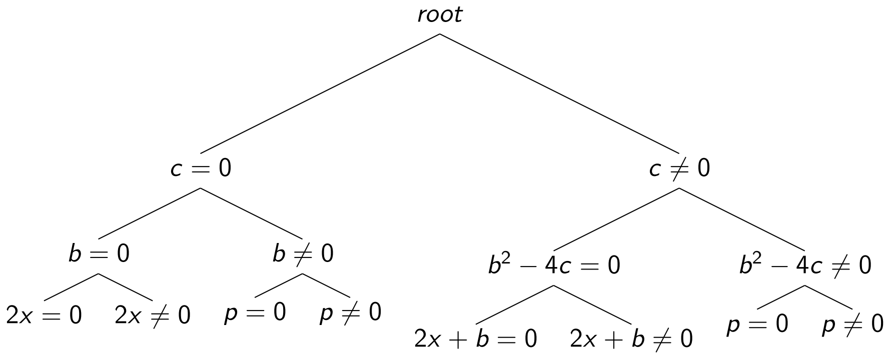


Figure: Complete complex cylindrical tree for the general monic quadratic equation, $p := x^2 + bx + c$, under variable ordering $c \prec b \prec x$.

Note that $b = 0$ is only tested where relevant

So how do I use these tools?

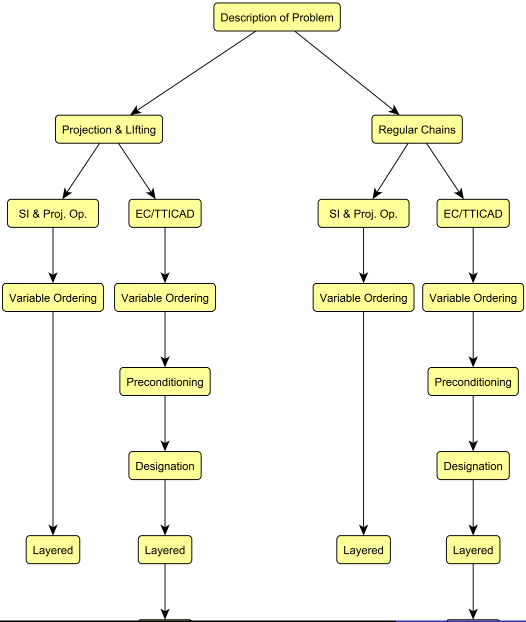
That's actually a very good question: there's a lot of choice in how to phrase the question

1. Choice of variable ordering (where permitted)
2. Choice of equalities
3. Choice of overall technology (Projection/Regular Chains/. . .)
4. Choice of how the problem is posed
5. (including Gröbner pre-conditioning)

- Choice of software: no software has (even close to) all the techniques, and each has extra "features"

These are **not** independent questions

How might this look? Wilson's thesis



Variable ordering

Theorem ([BD07])

There are CAD problems doubly exponential (in n) for all orderings, and other problems which are doubly exponential (in n) for some orderings, but constant for others

How to tell which case we're in?
How to choose the best (legal) ordering?
This was described in [HEW$^+$14]:
a variety of heuristics, with a machine-learning meta-heuristic

Bibliography

I

J.A. Abbott, R.J. Bradford, and J.H. Davenport.
A Remark on Factorisation.
SIGSAM Bulletin 2, 19:31–33, 1985.

C.W. Brown and J.H. Davenport.
The Complexity of Quantifier Elimination and Cylindrical Algebraic Decomposition.
In C.W. Brown, editor, *Proceedings ISSAC 2007*, pages 54–60, 2007.

R.J. Bradford, J.H. Davenport, M. England, S. McCallum, and D.J. Wilson.
Cylindrical Algebraic Decompositions for Boolean Combinations.
In *Proceedings ISSAC 2013*, pages 125–132, 2013.

Bibliography II

R.J. Bradford, J.H. Davenport, M. England, S. McCallum, and D.J. Wilson.
Truth Table Invariant Cylindrical Algebraic Decomposition.
To appear in *J. Symbolic Comp.*: http://arxiv.org/abs/1401.0645, 2014.

L. Busé and B. Mourrain.
Explicit factors of some iterated resultants and discriminants.
Math. Comp., 78:345–386, 2009.

C.W. Brown.
Improved Projection for Cylindrical Algebraic Decomposition.
J. Symbolic Comp., 32:447–465, 2001.

C. Chen, M. Moreno Maza, B. Xia, and L. Yang.
Computing Cylindrical Algebraic Decomposition via Triangular Decomposition.
In J. May, editor, *Proceedings ISSAC 2009*, pages 95–102, 2009.

G.E. Collins.
Quantifier Elimination for Real Closed Fields by Cylindrical Algebraic Decomposition.
In *Proceedings 2nd. GI Conference Automata Theory & Formal Languages*, pages 134–183, 1975.

J.H. Davenport and J. Carette.
The Sparsity Challenges.
In S. Watt *et al.*, editor, *Proceedings SYNASC 2009*, pages 3–7, 2010.

Bibliography IV

J.H. Davenport and J. Heintz.
Real Quantifier Elimination is Doubly Exponential.
J. Symbolic Comp., 5:29–35, 1988.

M. England, R. Bradford, C. Chen, J.H. Davenport, M.M. Maza, and D.J. Wilson.
Problem formulation for truth-table invariant cylindrical algebraic decomposition by incremental triangular decomposition.
In S.M. Watt *et al.*, editor, *Proceedings CICM 2014*, pages 45–60, 2014.

E. Fortuna, P. Gianni, and B. Trager.
Degree reduction under specialization.
J. Pure Appl. Algebra, 164:153–163, 2001.

P. Gianni.
Properties of Gröbner bases under specializations.
In *Proceedings EUROCAL 87*, pages 293–297, 1989.

Z. Huang, M. England, D. Wilson, J.H. Davenport, L.C. Paulson, and J. Bridge.
Applying machine learning to the problem of choosing a heuristic to select the variable ordering for cylindrical algebraic decomposition.
In S.M.Watt *et al.*, editor, *Proceedings CICM 2014*, pages 92–107, 2014.

H. Hong.
Improvements in CAD-Based Quantifier Elimination.
PhD thesis, OSU-CISRC-10/90-TR29 Ohio State University, 1990.

M. Kalkbrener.
Solving systems of algebraic equations by using Gröbner bases.
In *Proceedings EUROCAL 87*, pages 282–292, 1989.

S. McCallum.
An Improved Projection Operation for Cylindrical Algebraic Decomposition of Three-dimensional Space.
J. Symbolic Comp., 5:141–161, 1988.

S. McCallum.
On Projection in CAD-Based Quantifier Elimination with Equational Constraints.
In S. Dooley, editor, *Proceedings ISSAC '99*, pages 145–149, 1999.

M. Monagan and R. Pearce.
Sparse polynomial division using a heap.
J. Symbolic Comp., 46:807–822, 2011.

A. Schinzel.
On the greatest common divisor of two univariate polynomials, I.
In *A Panorama of number theory or the view from Baker's garden*, pages 337–352. C.U.P., 2003.

A. Tarski.
A Decision Method for Elementary Algebra and Geometry.
2nd ed., Univ. Cal. Press. Reprinted in *Quantifier Elimination and Cylindrical Algebraic Decomposition* (ed. B.F. Caviness & J.R. Johnson), Springer-Verlag, Wein-New York, 1998, pp. 24–84., 1951.

M. van Hoeij.
Factoring polynomials and the knapsack problem.
J. Number Theory, 95:167–189, 2002.

P.S. Wang.
An Improved Multivariable Polynomial Factorising Algorithm.
Math. Comp., 32:1215–1231, 1978.

IX

O. Zariski.
On equimultiple subvarieties of algebroid hypersurfaces.
Proc. Nat. Acad. Sci., 72:1425–1426, 3260, 1975.

R.E. Zippel.
Probabilistic Algorithms for Sparse Polynomials.
In *Proceedings EUROSAM 79*, pages 216–226, 1979.