Need Polynomial Systems Be Doubly-Exponential?

Davenport, J. H. & England, M.

Author post-print (accepted) deposited by Coventry University's Repository

Need Polynomial Systems be Doubly-exponential?

James H. Davenport[1] and Matthew England[2]

[1] Department of Computer Science,
University of Bath, Bath, BA2 7AY, UK
`J.H.Davenport@bath.ac.uk`,
WWW home page: `http://people.bath.ac.uk/masjhd/`

[2] School of Computing, Electronics & Maths,
Faculty of Engineering, Environment & Computing,
Coventry University, Coventry, CV1 5FB, UK
`Matthew.England@coventry.ac.uk`,
WWW home page: `http://computing.coventry.ac.uk/~mengland/`

Abstract. Polynomial Systems, or at least their algorithms, have the reputation of being doubly-exponential in the number of variables [Mayr and Mayer, 1982], [Davenport and Heintz, 1988]. Nevertheless, the Bezout bound tells us that that number of zeros of a zero-dimensional system is singly-exponential in the number of variables. How should this contradiction be reconciled?
We first note that [Mayr and Ritscher, 2013] shows that the doubly exponential nature of Gröbner bases is with respect to the dimension of the ideal, not the number of variables. This inspires us to consider what can be done for Cylindrical Algebraic Decomposition which produces a doubly-exponential number of polynomials of doubly-exponential degree. We review work from ISSAC 2015 which showed the number of polynomials could be restricted to doubly-exponential in the (complex) dimension using McCallum's theory of reduced projection in the presence of equational constraints. We then discuss preliminary results showing the same for the degree of those polynomials. The results are under primitivity assumptions whose importance we illustrate.

Keywords: computer algebra, cylindrical algebraic decomposition, equational constraint, Gröbner bases, quantifier elimination

1 Introduction

We consider the title question for two of the main tools for polynomial systems: *Gröbner Bases* (GB) and *Cylindrical Algebraic Decomposition* (CAD). For both the common claims of "doubly exponential", refers to "doubly exponential in the number of variables n". All other dependencies, on polynomial degrees d, polynomial coefficient length l, or number of polynomials m, are themselves polynomial in these quantities (albeit with the exponent of d and m possibly exponential in n).

In Section 2 we recall recent improvements to the analysis for GB which inspires us to revisit the complexity of CAD in Section 3. Here we describe how recent work for CAD in the presence of equational constraints (equations logically implied by the input) allows for a more subtle analysis. The progress is under the assumption of primitive equational constraints and in Section 4 we elaborate on the importance of this.

2 Gröbner Bases

A *Gröbner Basis* (GB) is a particular generating set of an ideal I (defined with respect to a monomial ordering). One definition is that the ideal generated by the leading terms of I is generated by the leading terms of the GB. GB theory allows properties of the ideal to be deduced such as dimension and number of zeros and so are one of the main practical tools for working with polynomial systems. Introduced by Buchberger in his PhD thesis of 1965 [10]; there has been much research to improve and optimise GB calculation, with the F_5 algorithm [21] perhaps the most used approach currently.

It is common (and the authors have done this, to write) "[31] shows that the computation of Gröbner bases is doubly exponential in the number of variables". It is unfortunately also common simply to write "[31] shows that the computation of Gröbner bases is doubly exponential", which while strictly correct if one counts the number of bits in a suitable encoding, is not particularly helpful.

However, we have known for a long time that the complexity of a Gröbner base of a zero-dimensional ideal is "only" singly-exponential in n [27]. These days, a much better reference is [32], which establishes both upper and lover bounds which are *singly* exponential in n, but *doubly* exponential in r, the actual dimension of the ideal. Clearly $r \leq n$, and only in the worst case is $r = n$.

Though we are currently unable to capitalise on the fact, we note that the examples of [31,32] are of non-radical ideals. The effective Nullstellensatz of [26] is only singly-exponential in the number of variables for membership in the *radical of* an ideal, giving us reason to believe it may be possible to prove a singly-exponential bound for radical ideals. GB technology is also needed to realise similar improvements to the complexity bound of CAD, as discussed next.

3 Cylindrical Algebraic Decomposition

3.1 Background

A *cylindrical algebraic decomposition* (CAD) is a *decomposition* of $\mathbb{R}^n$ into cells. The cells are arranged *cylindrically*, meaning the projections of any pair with respect to the given ordering are either equal or disjoint. We assume variables labelled according to their ordering (so the projections considered are $(x_1, \ldots, x_\ell) \to (x_1, \ldots, x_k)$ for $k < \ell$) with the highest ordered variable present said to be the *main variable*. Finally, by *algebraic* we mean semi-algebraic: each cell can be described with a finite sequence of polynomial constraints.

A CAD is produced to be invariant for input; originally *sign-invariant* for a set of input polynomials (so on each cell each polynomial is positive, zero or negative), and more recently *truth-invariant* for input Boolean-valued formulae built from the polynomials (so on each cell each formula is either true or false). Unlike Gröbner Bases we may now consider general polynomial systems instead of just equations.

CAD usually involves two phases. The first *projection*, applies operators recursively on polynomials, each time producing a set with one less variable which together define the *projection polynomials*. These are used in the second phase, *lifting*, to build CADs incrementally by dimension. First a CAD of the real line is built according to the real roots of the univariate polynomials. Next, a CAD of $\mathbb{R}^2$ is built by repeating the process over each cell in $\mathbb{R}^1$ with the bivariate polynomials evaluated at a sample point of the cell in $\mathbb{R}^1$. We call the cells where a polynomial vanishes *sections* and those regions in-between *sectors*, which together form the *stack* over the cell. Taking the union of these stacks gives the CAD of $\mathbb{R}^2$. The process is repeated until a CAD of $\mathbb{R}^n$ is produced. In each lift we extrapolate the conclusions drawn from working at a sample point to the whole cell requiring validity theorems for the projection operator used.

CAD was originally introduced by Collins for quantifier elimination (QE) in real closed fields [1] with applications since ranging from parametric optimisation [22] and epidemic modelling [9], to reasoning with multi-valued functions [15] and the derivation of optimal numerical schemes [20]. There has been much work on improving Collins' original approach most notably refinements to the projection operator [28] [6], [23]; early termination of lifting [14] [35]; and symbolic-numeric schemes [33], [25]. Some recent advances include dealing with multiple formulae [3], [4]; local projection [7], [34]; decompositions via complex space [12], [2]; and the development of heuristics for CAD problem formulation [5], [17], [36] including machine learned approaches [24].

3.2 Complexity

CAD has long been known to have worst case complexity doubly exponential [8],[16]. Suppose the input consists of m polynomials (perhaps derived from formulae) in n variables of maximum degree d in any one variable. Section 2.3 of [4] describes in detail how the complexity of CAD algorithms may be measured in terms of a bound on the total number of cells produced (closely correlated to the timings but allowing for simpler implementation independent comparisons) based on improvements to techniques introduced by McCallum's thesis. In particular, the dominant term in that bound for a sign-invariant CAD produced using the algorithm of [28] is

$$(2d)^{2^n-1}m^{2^n-1}2^{2^{n-1}-1}. \tag{1}$$

I.e. the CAD grows doubly exponentially with the number of variables n. The analysis shows that by the end of the projection stage we have M polynomials in $\mathbb{R}^1$, each of degree D, where $D = d^{2^{O(n)}}$ and $M = m^{2^{O(n)}}$. However, [16] [8]

respectively find lower bounds with $D = d^{2^{\Omega(n)}}$ and $M = m^{2^{\Omega(n)}}$ with the underlying polynomials all simple, showing that the doubly-exponential difficulty of CAD resides in *the complicated number of ways simple polynomials can interact.*

So need CAD be doubly exponential? Given the previous discussion the answer is yes, but as with GB we need not settle for "doubly exponential *in the number of variables n*". We might hope for "doubly exponential *in the dimension*", but this is thwarted by the fact that the examples of [8],[16] are in fact zero-dimensional. Nevertheless, we can take advantage of certain dimensional reductions when made explicit through the identification of *equational constraints* (ECs), polynomial equations logically implied by formulae.

The presence of an EC restricts the dimension of the solution space and so we may expect the CAD to be doubly exponential in $n - \ell$ where ℓ is the number of ECs taken advantage of. Of course, we would no longer be building CADs sign-invariant for polynomials but ones truth-invariant for formulae. The present authors have demonstrated this first for the part of the bound dependent on m (number of polynomials) in [18] and then for the part dependent on d (maximum degree) in [19] (work currently submitted for publication).

3.3 CAD with multiple ECs

Collins noticed that in the presence of an EC a truth-invariant CAD need only be sign-invariant throughout for the defining polynomial of the EC with other polynomials sign-invariant only on the sections of that polynomial [13]. This led McCallum to develop restrictions to his projection operator from [28] in [29] (for the first projection) and [30] (for subsequent projections). See [18, Section 2.1] for a more detailed summary of this theory. These operators work with a single EC and so the CAD algorithm may take advantage of only one in each main variable. However, [30] also introduced a process to derive ECs in lower main variables based on the observation that the resultant of the polynomials defining two ECs itself defines an EC.

In [18] the present authors reviewed the theory of reduced projection operators. In particular we introduced two refinements to the lifting phase of CAD which follow from McCallum's theory of reduced projection operators:

1. Minimising lifting polynomials: When lifting *to* $\mathbb{R}^k$ if there exists an EC with main variable k then we need only lift with respect to (isolate roots of) this.
2. Minimising real root isolation: When lifting *over* $\mathbb{R}^k$ if there exists an EC with main variable k then we need only isolate real roots over sections (allowing sectors to be trivially lifted to a cylinder).

These refinements require us to discard two embedded principles of CAD:

- That the projection polynomials are a fixed set: we now differ the polynomials used in projection from lifting and keep track of which relate to ECs.

– That the invariance structure of the final CAD can be expressed in terms of sign-invariance of polynomials: The final CAD may not no longer be sign-invariant for any one polynomial polynomials, even ECs, but is still guaranteed to be truth invariant for the formula.

In [18, Section 5] we used the complexity analysis techniques of [4] to show that a CAD in which the first ℓ projections had a designated EC had dominant term complexity bound of the form $(2d)^{\mathcal{O}(2^n)}(2m)^{\mathcal{O}(2^{n-\ell})}$. I.e. we have reduced the number of polynomials involved accordingly but not their degree.

The present authors considered what could be done with respect to the degree recently in [19]. The theory of iterated resultants as considered by Busé and Mourrain [11] suggested that the iterated univariate resultants produced by CAD (and in particular in the identification of ECs for subsequent projections) were more complicated that the information they needed to encode. The true multivariate resultants were contained as a factor and grow in degree exponentially rather than doubly exponentially. The key result had to be adapted from [11] to change the arguments from total degree in all variables to the degree in at most one variable required for bounding the number of CAD cells produced.

The authors proposed using GB technology for the generation of the ECs in subsequent projections to realise this limit in degree growth. This leads to the other projection polynomials growing exponentially in $O(\ell^2)$ but remember that these are not used during lifting ([18] improvement (1) from above) and thus not counted towards the cell count bounds (although they do boost the degree of polynomials involved in projections without ECs). The outcome of this approach is a dominant term complexity bound of the form $(\ell d)^{\mathcal{O}(2^{n-\ell})}(2m)^{\mathcal{O}(2^{n-\ell})}$.

Restrictions There are some restrictions to the work as acknowledged in [18] and [19]. First, the analysis assumes the designated ECs are in strict succession at the start of projection. This restrictions was made to ease the complexity analysis (with the formal algorithm specification and implementations not adhering).

The substantial restriction is that the theory of CAD with multiple ECs is only developed for primitive ECs. Possibilities to remove this restriction are discussed in [18] and could involve leveraging the TTICAD theory of [3] [4]. A TTICAD (truth-table invariant CAD) allows for savings from ECs when building a CAD for multiple formulae at once. Currently the theory is only developed for ECs in the main variable of the system and so an analogous extension to subsequent projections is first required for TTICAD itself.

4 The primitivity restriction

We finish by considering the classic complexity results of [8],[16] in light of the above recent progress. We see the importance of the aforementioned primitivity restriction.

The examples in both [8] and [16] rest on the following construction. Let $P_k(x_k, y_k)$ be the statement $x_k = f(y_k)$ and then define recursively

$$P_{k-1}(x_{k-1}, y_{k-1}) := \tag{2}$$

$$\underbrace{\exists z_k \forall x_k \forall y_k}_{Q_k} \underbrace{((y_{k-1} = y_k \wedge x_k = z_k) \vee (y_k = z_k \wedge x_{k-1} = x_k))}_{L_k} \Rightarrow P_k(x_k, y_k).$$

This is $\exists z_k \, (z_k = f(y_{k-1}) \wedge x_{k-1} = f(z_k))$, i.e. $x_{k-1} = f(f(y_{k-1}))$. It is repeated nesting of this procedure that builds the doubly-exponential growth, so that

$$Q_{k-1} L_{k-1} \Rightarrow (Q_k L_k \Rightarrow P_k(x_k, y_k)), \tag{3}$$

gives $x_{k-2} = f(f(f(f(y_{k-2}))))$ etc. Rewriting (3) in prenex form gives

$$Q_{k-1} Q_k \neg L_{k-1} \vee \neg L_k \vee P_k(x_k, y_k). \tag{4}$$

The negation of (4) is therefore

$$\overline{Q}_{k-1} \overline{Q}_k L_{k-1} \wedge L_k \wedge \neg P_k(x_k, y_k), \tag{5}$$

where the $\overline{}$ operator interchanges $\forall$ and $\exists$.

Now, L_k can be rewritten as

$$L_k = (y_{k-1} = y_k \vee y_k = z_k) \wedge (y_{k-1} = y_k \vee x_{k-1} = x_k)$$
$$\wedge (x_k = z_k \vee y_k = z_k) \wedge (x_k = z_k \vee x_{k-1} = x_k) \tag{6}$$

and further

$$L_k = (y_{k-1} - y_k)(y_k - z_k) = 0 \wedge (y_{k-1} - y_k)(x_{k-1} - x_k) = 0$$
$$\wedge (x_k - z_k)(y_k - z_k) = 0 \wedge (x_k - z_k)(x_{k-1} - x_k) = 0, \tag{7}$$

which shows L_k to be a conjunction of (imprimitive) equational constraints. This is true for any L_i, hence the propositional part of (5) is a conjunction of eight equalities, mostly imprimitive, and $\neg P_k(x_k, y_k)$. Furthermore there are equalities whose main variables are the first variables to be projected if we try to produce a quantifier-free form of (5). But the quantifier-free form of (5) describes the complement of the semi-algebraic varieties in [8] or [16] (depending which P_k we take) and these have doubly-exponential complexity in n.

The discussion of this section shows the relevance of the primitivity restriction discussed at the end of the previous section and imposed in the work of [18], [19]. It may be more than a technicality to remove them.

References

1. D. Arnon, G.E. Collins, and S. McCallum. Cylindrical algebraic decomposition I: The basic algorithm. *SIAM Journal of Computing*, 13:865–877, 1984.
2. R. Bradford, C. Chen, J.H. Davenport, M. England, M. Moreno Maza, and D. Wilson. Truth table invariant cylindrical algebraic decomposition by regular chains. In *Computer Algebra in Scientific Computing* (LNCS 8660), p.44–58. Springer, 2014.
3. R. Bradford, J.H. Davenport, M. England, S. McCallum, and D. Wilson. Cylindrical algebraic decompositions for Boolean combinations. In Proc. ISSAC '13, pp. 125–132. ACM, 2013.
4. R. Bradford, J.H. Davenport, M. England, S. McCallum, and D. Wilson. Truth table invariant cylindrical algebraic decomposition. *Journal of Symbolic Computation*, 76:1–35, 2015.
5. R. Bradford, J.H. Davenport, M. England, and D. Wilson. Optimising problem formulations for cylindrical algebraic decomposition. In *Intelligent Computer Mathematics* (LNCS 7961), pages 19–34. Springer Berlin Heidelberg, 2013.
6. C.W. Brown. Improved projection for cylindrical algebraic decomposition. *Journal of Symbolic Computation*, 32(5):447–465, 2001.
7. C.W. Brown. Constructing a single open cell in a cylindrical algebraic decomposition. In Proc. ISSAC '13, pages 133–140. ACM, 2013.
8. C.W. Brown and J.H. Davenport. The complexity of quantifier elimination and cylindrical algebraic decomposition. In Proc. ISSAC '07, pages 54–60. ACM, 2007.
9. C.W. Brown, M. El Kahoui, D. Novotni, and A. Weber. Algorithmic methods for investigating equilibria in epidemic modelling. *Journal of Symbolic Computation*, 41:1157–1173, 2006.
10. B. Buchberger. Bruno Buchberger's PhD thesis (1965): An algorithm for finding the basis elements of the residue class ring of a zero dimensional polynomial ideal. *Journal of Symbolic Computation*, 41(3-4):475–511, 2006.
11. L. Busé and B. Mourrain. Explicit factors of some iterated resultants and discriminants. *Mathematics of Computation*, 78:345–386, 2009.
12. C. Chen, M. Moreno Maza, B. Xia, and L. Yang. Computing cylindrical algebraic decomposition via triangular decomposition. In Proc. ISSAC '09, pages 95–102. ACM, 2009.
13. G.E. Collins. Quantifier elimination by cylindrical algebraic decomposition – 20 years of progress. In B. Caviness and J. Johnson, editors, *Quantifier Elimination and Cylindrical Algebraic Decomposition*, Texts & Monographs in Symbolic Computation, pages 8–23. Springer-Verlag, 1998.
14. G.E. Collins and H. Hong. Partial cylindrical algebraic decomposition for quantifier elimination. *Journal of Symbolic Computation*, 12:299–328, 1991.
15. J.H. Davenport, R. Bradford, M. England, and D. Wilson. Program verification in the presence of complex numbers, functions with branch cuts etc. In *Proc. SYNASC '12*, pages 83–88. IEEE, 2012.
16. J.H. Davenport and J. Heintz. Real quantifier elimination is doubly exponential. *Journal of Symbolic Computation*, 5(1-2):29–35, 1988.
17. M. England, R. Bradford, C. Chen, J.H. Davenport, M. Moreno Maza, and D. Wilson. Problem formulation for truth-table invariant cylindrical algebraic decomposition by incremental triangular decomposition. In *Intelligent Computer Mathematics*, (LNAI 8543), pages 45–60. Springer International, 2014.
18. M. England, R. Bradford, and J.H. Davenport. Improving the use of equational constraints in cylindrical algebraic decomposition. In *Proc. ISSAC '15*, pages 165–172. ACM, 2015.

19. M. England and J.H. Davenport. The complexity of cylindrical algebraic decomposition with respect to polynomial degree. *Submitted for Publication*, 2016. Preprint available at `http://arxiv.org/abs/1605.02494`

20. M. Erascu and H. Hong. Synthesis of optimal numerical algorithms using real quantifier elimination (Case Study: Square root computation). In *Proc. ISSAC '14*, pages 162–169. ACM, 2014.

21. J.C. Faugère. A new efficient algorithm for computing Gröbner bases without reduction to zero (F5). In *Proc. ISSAC '02*, pages 75–83. ACM, 2002.

22. I.A. Fotiou, P.A. Parrilo, and M. Morari. Nonlinear parametric optimization using cylindrical algebraic decomposition. In *Decision and Control, 2005 European Control Conference. CDC-ECC '05.*, pages 3735–3740, 2005.

23. J. Han, L. Dai, and B. Xia. Constructing fewer open cells by gcd computation in CAD projection. In *Proc. ISSAC '14*, pages 240–247. ACM, 2014.

24. Z. Huang, M. England, D. Wilson, J.H. Davenport, L. Paulson, and J. Bridge. Applying machine learning to the problem of choosing a heuristic to select the variable ordering for cylindrical algebraic decomposition. In *Intelligent Computer Mathematics* (LNAI 8543), pages 92–107. Springer International, 2014.

25. H. Iwane, H. Yanami, H. Anai, and K. Yokoyama. An effective implementation of a symbolic-numeric cylindrical algebraic decomposition for quantifier elimination. In *Proc. SNC '09*, pages 55–64, 2009.

26. J. Kollár. Sharp effective Nullstellensatz. *Journal of the American Mathematical Society*, 1:963–975, 1988.

27. D. Lazard. Gröbner Bases, Gaussian elimination and resolution of systems of algebraic equations. In *Proc. EUROCAL 83*, pages 146–157, 1983.

28. S. McCallum. An improved projection operation for cylindrical algebraic decomposition. In B. Caviness and J. Johnson, editors, *Quantifier Elimination and Cylindrical Algebraic Decomposition*, Texts & Monographs in Symbolic Computation, pages 242–268. Springer-Verlag, 1998.

29. S. McCallum. Factors of iterated resultants and discriminants. *Journal of Symbolic Computation*, 27(4):367–385, 1999.

30. S. McCallum. On propagation of equational constraints in CAD-based quantifier elimination. In *Proc. ISSAC '01*, pages 223–231. ACM, 2001.

31. E.W. Mayr and A.R. Meyer. The complexity of the word problems for commutative semigroups and polynomial ideals. *Advances in Mathematics*, 46(3):305–329, 1982.

32. E.W. Mayr and S. Ritscher. Dimension-dependent bounds for Gröbner bases of polynomial ideals. *Journal of Symbolic Computation*, 49:78–94, 2013.

33. A. Strzeboński. Cylindrical algebraic decomposition using validated numerics. *Journal of Symbolic Computation*, 41(9):1021–1038, 2006.

34. A. Strzeboński. Cylindrical algebraic decomposition using local projections. In *Proc. ISSAC '14*, pages 389–396. ACM, 2014.

35. D. Wilson, R. Bradford, J.H. Davenport, and M. England. Cylindrical algebraic sub-decompositions. *Mathematics in Computer Science*, 8:263–288, 2014.

36. D. Wilson, M. England, J.H. Davenport, and R. Bradford. Using the distribution of cells by dimension in a cylindrical algebraic decomposition. In *Proc. SYNASC '14*, pages 53–60. IEEE, 2014.