Varieties of Doubly-Exponential Behaviour in Cylindrical Algebraic Decomposition

James H. Davenport[1]

[1]University of Bath, Faculty of Science, Department of Computer Science, Bath, UK

Abstract

It is almost fifty years since cylindrical algebraic decomposition was introduced: it was far better than previous ideas, but the algorithm was doubly exponential in the number of variables. Various mitigations have been developed over the last forty years. But we have known for over thirty years that cylindrical algebraic decomposition has a worst-case lower bound doubly-exponential in the number of quantifier alternations, which in worst case is proportional to the number of variables. This lower bound can describe the degree of the polynomials, or the number of polynomials, or both. This paper explores the reasons for this, and what further developments, theoretical or practical, might be possible.

Keywords

Quantifier Elimination, Cylindrical Algebraic Decomposition, Equational Constraints

1. Introduction

Cylindrical Algebraic Decomposition was introduced as the first practical, albeit doubly exponential in n the number of variables, tool to solve Real Quantifier Elimination in [1]. Since then there have been many developments in the algorithms: see Table 1. Let d be the total degree of the input polynomials, m the number of input polynomials, q the number of equational constraints, n the number of variables and a the number of alternations of quantifiers, so that $a \le n-1$.

Table 1

Summary table: e_d and e_m are double exponents of d and m: $d^{2^{e_d}} m^{2^{e_m}}$

Idea	e_m	e_d
Collins (see Appendix A)	$n+O(1)$	$(\log_2 3)n+O(1)$
McCallum (but nullification)	$n+O(1)$	$n+O(1)$
Lazard (proof in [2])	$n+O(1)$	$n+O(1)$
Equational Constraints	(?) $n-q+O(1)$	(?) $n-q+O(1)$
Virtual Term Substitution	(?) $O(a)$	challenges
Comprehensive Gröbner Bases	(??) $O(a)$	(??) $O(a)-n+O(1)$
Regular Chains	(??) $n-q+O(1)$	(??) $n-q+O(1)$

Here "(?)" means "under suitable conditions" (not always very well-defined), and "(??)" means "wild guess by the author": see later.

SC^2 *2021: 6th International Workshop on Satisfiability Checking and Symbolic Computation, August 19–20, 2021*
J.H.Davenport@bath.ac.uk (J. H. Davenport)
0000-0002-3982-7545 (J. H. Davenport)

It is thirty five years since Davenport & Heintz sat in a café in Strasbourg and wrote the draft of "Real Quantifier Elimination is Doubly Exponential". "Doubly exponential" means that the complexity is $d^{2^{e_d}} m^{2^{e_m}}$ where e_d and e_m depend non-trivially on n (or on a).

Theorem 1. $e_d, e_m \in \Omega(n)$. *More precisely, Davenport & Heintz [3] showed that, for any algorithm solving quantifier elimination, whether or not by constructing cylindrical algebraic decompositions (CAD), both* e_d *and* e_m *were at least* $n/5 + O(1)$*, with* a *being* $\Theta(n)$ *(in fact* $2n/5+O(1)$*), and Brown & Davenport [4] showed (again with* a *being* $\Theta(n)$*, this time* $2n/3+O(1)$*) that* e_m *was at least* $n/3 + O(1)$*, even if* $d = 1$*.*

Collins' initial CAD construction [1] to solve quantifier elimination had an upper bound for the double exponents of $(\log_2 3)n + O(1)$, reduced (conditional on no nullification) to $n + O(1)$ by McCallum [5], and unconditionally by the ideas of Lazard [6, justified by [2]]. McCallum could reduce e_m when there were equational constraints (and no nullification), but there are problems in translating this to the Lazard setting [7]. Davenport & England [8, 9] can use equational constraints and Gröbner bases to reduce e_d, but again there are conditions.

2. Collins' algorithm and its descendants

We first consider the Collins algorithm and its descendants. These are generically known as "Projection and Lifting" algorithms. Some other approaches to CAD or to real quantifier elimination will be discussed in subsequent sections.

2.1. Projection Polynomials

The obvious problems with real algebraic geometry in two dimensions x and y are that two curves (zeros of $f(x, y)$ and $g(x, y)$) can cross, or that a curve can bend back on itself, or go through infinity. We can detect the x coordinates (projections on the x axis) of such potential trouble easily enough, by producing "projection polynomials" in x alone.

The resultant $\mathrm{res}_y(f, g)$ is a polynomial in x, whose roots are the values of x above which f and g cross. Note that it is possible that they may cross when x is real but y is not, but this is a problem we would discover later on in the lifting process: see Question 3.

The discriminant $\mathrm{disc}_y(f)$ is a polynomial in x, whose roots are the values of x above which f is momentarily vertical, and so *may* double back on itself. Again, it might do this in complex space, but again this is a problem we would discover later on in the lifting process.

The leading coefficient $\mathrm{lc}_y(f)$ (with respect to y) is a polynomial whose roots are the values of x above which f is momentarily infinite. The same caveat about complex values applies here.

The same applies in n dimensions $x_1, \ldots, x_n$, and we have to consider $\mathrm{res}_{x_n}(f, g)$, $\mathrm{disc}_{x_n}(f)$ and $\mathrm{lc}_{x_n}(f)$, which are polynomials in $x_1, \ldots x_{n-1}$, collectively known as the *projection* of the original polynomials.

2.2. Lifting and Nullification

The trouble is that, on some subspace of $\mathbf{R}^{n-1}$, one of these projection polynomials may vanish identically (be *nullified*), and, while telling us that there are problems here, this may conceal the fact that there are multiple kinds of problems. The solution for the nullification of a leading coefficient is to consider more coefficients in the projection phase. Conceptually we consider enough coefficients that we can be sure that they do not all vanish simultaneously: from the point of view of complexity analysis we bound this by considering all coefficients.

Nullification of a resultant or discriminant is more tedious, and we have to consider, not just them, but all the principal subresultants (see, e.g. [10]) on the way to computing them. If our polynomials have degree d, there may be $\Theta(d)$ such subresultants for one resultant or discriminant, and it is this that accounts for the relatively worse complexity of Collins' method.

2.3. McCallum's improvement

Collins considered *sign-invariant* polynomials, i.e. in every region of the cylindrical algebraic decomposition, each polynomial must be uniformly positive, or negative, or zero. Since a polynomial cannot go from positive to negative except via zero, we are really looking at vanishing/non-vanishing. McCallum [5] strengthened this to *order-invariant*, i.e. we insist on the same order of vanishing. He also did not consider the subresultants. This gave him $e_m, e_d = n + O(1)$, at the cost of not handling nullification. This gave rise to various developments.

- [5] Lift *order-invariant* polynomials: $2^{2^{n+O(1)}}$.
- **But** we give up (i.e. revert to [1]) if any polynomial becomes identically zero over any region, e.g. $(x^2 + y^2)z + (x^4 + y^4)$ over $x = y = 0$: these are *nullifying regions*.
- [11] If we have $\Phi := P(y_1, \ldots, y_n) = 0 \land \hat{\Phi}$ (equational constraint). Reduces n by 1 in e_m.
- [12] Several s equational constraints. Reduces n by s in e_m.
- [13] The equational constraints don't need to be $\land$ with the rest: consider Φ as a truth table.
- [14] Shows that s equational constraints reduce n by s in both e_m and e_d, *if the relevant projection polynomials are primitive*, necessary by [8]. See [15].
- [2] Lift *Lex-least invariant* polynomials (idea from [6], flawed proof) — gets rid of the **But** issue with [5] and slightly improves the complexity.
- [16] Improvement to [2]: doesn't change asymptotic complexity but useful in practice.

Challenge: can we merge [2, 16] with equational constraints, either [11] or [12], or [13] or even [15]?

3. Equational Constraints

Consider $f(y_1, \ldots, y_n) = 0 \land (g_1 > 0 * g_2 > 0)$ (and in general k g_i), where $*$ is either $\land$ or $\lor$.

- [5] To understand $\mathbf{R}^n$ we project $\mathrm{disc}_{y_n}(f), \mathrm{disc}_{y_n}(g_1), \mathrm{disc}_{y_n}(g_2), \mathrm{res}_{y_n}(f, g_1), \mathrm{res}_{y_n}(f, g_2)$, $\mathrm{res}_{y_n}(g_1, g_2)$ (and in general $k(k+3)/2$ polynomials), assuming none of these have nullifying regions.

[11] To understand $\mathbf{R}^n|_{f=0}$ we project just $\operatorname{disc}_{y_n}(f), \operatorname{res}_{y_n}(f, g_1), \operatorname{res}_{y_n}(f, g_2)$ (and in general $k+1$ polynomials). In the absence of nullification this is sufficient: for example $\operatorname{res}_{y_{n-1}}(\operatorname{res}_{y_n}(f, g_1), \operatorname{res}_{y_n}(f, g_2))$ contains all the information we need from $\operatorname{res}_{y_n}(g_1, g_2)$, and $\operatorname{disc}_{y_{n-1}}(\operatorname{res}_{y_n}(f, g_i))$ contains all the information we need from $\operatorname{disc}_{y_n}(g_i)$, as we are only interested in their intersection with $f = 0$.

3.1. One Equational Constraint and Lex-least

The details of this challenge are in Akshar Nair's thesis [7]. The "obvious" merger is true: [17]. If there are no nullifying regions, then [11] transfers to [2] (and presumably [16], but this hasn't been formally proved). But if $\operatorname{res}_{y_{n-1}}(\operatorname{res}_{y_n}(f, g_1), \operatorname{res}_{y_n}(f, g_2))$ nullifies on a region (the foot of the curtain), we can no longer infer what g_1 and g_2 do on the curtain. The first task [18] is to detect the curtains, and determine if the foot is zero-dimensional or not. If the foot is zero-dimensional, we can still use equational constraint methodology. If the foot is not zero-dimensional, we can always revert to the original projection without considering equational constraints, a good solution is still an open problem.

3.2. Multiple Equational Constraints

[12] points out that, if we have $f_1(y_1, \ldots, y_n) = 0 \wedge f_2(y_1, \ldots, y_n) = 0 \wedge (g_1 > 0 * g_2 > 0)$, and we apply the techniques of §3 with f_1 as the equational constraint (and treating f_2 as a g_i), then in $y_1, \ldots, y_{n-1}$, $\operatorname{res}_{y_n}(f_1, f_2)$ is still an equational constraint. Since [11] lifted *order*-invariant decompositions to *sign*-invariant decompositions, we cannot nest this directly, but [12] adjusts the projection process and solves this difficulty.

4. Virtual Term Substitution

This idea, generally abbreviated to VTS, was introduced by Weispfenning in [19] for linear problems. Here $\cdots Q_n y_n \Phi(y_1, \ldots, y_n)$ in which y_n occurs linearly can be replaced by $\cdots \hat{\Phi}(y_1, \ldots, y_{n-1})$. This was extended in [20, 21] to the quadratic case and beyond, with details of the cubic case being in [22]. An extension to unbounded degree is given in [23], but the author knows of no public implementation, and this seems to be limited to univariate problems (i.e. no parameters), so we pass over it for the moment, though it is a suitable subject for further research.

A crude description would be "substituting in the critical values and their neighbours", but the details are more subtle, hence Weispfenning's concept of *virtual* term substitution.

In particular, if y_n occurs quadratically in $ay_n^2 + by_n + c$, with corresponding critical values $y_n = \frac{1}{2a}\left(-b \pm \sqrt{b^2 - 4ac}\right)$, there might be 0, 1 or 2 critical values, and we also need to worry about the case $a = 0$: hence VTS has substitutions with guards, and the result of eliminating an $\exists$ quantifier, and hence a block of $\exists$, is a disjunction, often large. However, VTS treats $\forall$ as $\neg_1\exists\neg_2$, so $\neg_2$ turns the disjunction into a conjunction, processing the $\exists$ builds a further disjunction on top of this, which $\neg_1$ turns back into a conjunction. Each of these conversions could have exponential blowup. Hence *provided we remain within the scope of VTS*, we might expect to have (this is a long way from being a proof!) a process which is doubly exponential in the number of alternations, but only singly exponential in the number of variables.

The details of work at Bath are in Zak Tonks' thesis [24]. As described in [25], he has implemented a poly-algorithm that does linear/quadratic VTS where feasible, and reverts to [2] (this is also the first known implementation of [2]) when not. The cubic case is also implemented, after some elaboration of the details in [22]. As described in [26], this implementation is "SMT-friendly", in the sense of supporting adding and retracting F_i, i.e. interfacing with the backtracking nature of DPLL(T) solvers.

5. Comprehensive Gröbner Systems

This method was also introduced by Weispfenning, in [27], and has a recent exploration in [28]. The key idea is this. We consider an "innermost block" in this form:

$$\exists \overline{x} \left(\begin{array}{c} f_1(\overline{y},\overline{x}) = 0 \wedge \cdots f_r(\overline{y},\overline{x}) = 0 \wedge \\ p_1(\overline{y},\overline{x}) > 0 \wedge \cdots p_s(\overline{y},\overline{x}) > 0 \wedge \\ q_1(\overline{y},\overline{x}) \neq 0 \wedge \cdots q_t(\overline{y},\overline{x}) \neq 0 \end{array} \right) \tag{1}$$

where $\overline{y}$ represents the remaining variables, and $f_i, p_j, q_k \in \mathbf{Q}[\overline{y},\overline{x}] \setminus \mathbf{Q}[\overline{y}]$. We introduce new variables $\overline{z}$ and $\overline{w}$, with $\overline{z}, \overline{w} \succ \overline{x}$, and consider the polynomials

$$\{f_1,\ldots,f_r, \underbrace{z_1^2 p_1 - 1,\ldots,z_s^2 p_s - 1}_{\text{forcing positive}}, \underbrace{w_1 q_1 - 1,\ldots,w_t q_t - 1}_{\text{forcing nonzero}}\}. \tag{2}$$

Let $\mathcal{G} = (S_i, G_i)_{i\in I}$ be a Comprehensive Gröbner System (with parameters $\overline{y}$) for (2) so that $\overline{y}$ space is partitioned by the S_i. Then the truth of (1) is equivalent to the truth of

$$\bigvee_{i\in I} \left(\Phi(S_i) \wedge \Psi(G_i)\right), \tag{3}$$

where $\Phi(S_i)$ is the defining formula for S_i and $\Psi(G_i)$ is the condition for G_i to have real roots, and hence (by (2)) for (1) to be satisfied. The derivation of $\Psi(G_i)$ from G_i is given in [28], and uses [29] to derive conditions for the G_i to have real roots.

Like VTS, this method treats $\forall$ as $\neg_1\exists\neg_2$, so $\neg_2$ turns the disjunction in (3) into a conjunction, processing the $\exists$ builds a further disjunction on top of this, which $\neg_1$ turns back into a conjunction. Hence again we might expect doubly exponential behaviour in the number of alternations, and *possibly* only singly-exponential in the number of variables. This would require the Comprehensive Gröbner basis computations to have that property, and this is not obvious (see [30, §5]).

Question 1. *What can we say about the complexity of Comprehensive Gröbner Systems-based methods?*

6. Regular Chains

The Regular Chains method is a fundamentally different way of solving polynomial systems than Gröbner bases: in particular it proceeds variable-by-variable: see [31]. Their complexity is discussed in [32], from which we take the following.

Definition 1. *The Gallo–Mishra degree of a polynomial* $f \in K[x_1, \ldots, x_n]$, $\deg_{\rm GM}(f)$, *is* $\sum_i \deg_{x_i}(f)$.

We have $\deg_{\rm total}(f) \le \deg_{\rm GM}(f) \le n \deg_{\rm total}(f)$. When $f = x_1^k + \cdots + x_n^k$, $\deg_{\rm GM}(f) = nk$ but $\deg_{\rm total}(f) = k$.

Definition 2. *Let* I *be an ideal in* $K[x_1, \ldots, x_n]$, *and* $\mathrm{TVar}(I)$ *be a maximal set of independent variables* $x_{i_1}, \ldots, x_{i_r}$, *i.e.* $I \cap K[x_{i_1}, \ldots, x_{i_r}] = \{0\}$. *Let* $\mathrm{AlgVar}(I)$ *be the remaining* x_i.

Notation 1 (Gallo–Mishra Assumption). *Assume, after renumbering if necessary, that* $\mathrm{AlgVar}(I) = \{x_{l+1}, \ldots, x_n\}$, *and that we have an ordering with the non-algebraic variables before the algebraic ones.*

Theorem 2 ([32, Theorem 3.4]). *Let* $I = (f_1, \ldots, f_s)$ *be an ideal in* $K[x_1, \ldots, x_n]$, *and* $\deg_{\rm GM}(f_i) \le d$. *Then, under Assumption 1,* I *has a characteristic set* $G = (g_1, \ldots, g_r)$ *where:*

1. $\mathrm{mvar}(g_j) = x_{j+l}$;
2. $\deg_{\rm GM}(g_j) \le 4(s+1)(9r)^{2r} d(d+1)^{4r^2}$;
3. $g_j = \sum a_{i,j} f_i$ *where* $\deg_{\rm GM}(a_{i,j} f_i) \le 11(s+1)(9r)^{2r} d(d+1)^{4r^2}$.

This theorem tells us that *for this order*, the degree is "only" singly-exponential, albeit $O(r^2) = O(n^2)$. It is less useful than it might seem, for it supposes that we know one of the options for $\mathrm{AlgVar}(T)$ before we start the process. In reality, we may not even know $|\mathrm{AlgVar}(T)|$. [32] refers to [33], but that deals with unmixed ideals (and we may not know that in advance) and is exponential with $O(n^2)$ as the exponent, rather than $O(r^2)$.

Regular Chains can be used to produce, first a complex cylindrical tree, and then a cylindrical algebraic decomposition: see [34] for the construction of Cylindrical Algebraic Decompositions and [35, 36] for Quantifier Elimination. The complexity of these translations from complex cylindrical trees has not been studied, to the best of the author's knowledge. However, it is (at least in the worst case) bad, since Theorem 2 has only a singly-exponential complexity, and Theorem 1 shows *Real* QE/CAD have doubly-exponential complexity.

Question 2. *What can we say about the complexity of Regular Chains-based methods?*

The paper [37] shows that the theory of equational constraints, which [38] extends to partial equational constraints, can be adapted to the Regular Chains approach, with significant gains in practice.

7. Next steps

The author is part of a joint Bath/Coventry project [39] to explore this area further.

Question 3. *In terms of practical efficiency gains, we said in §2.1 that* $\text{res}_y(f, g)$ *might have a real root* x_0*, but the corresponding* y *values might be complex. We might therefore want to discard* x_0*, but an implementation challenge is knowing there are no other reasons for considering* x_0*.*

Also, as part of a wider collaboration, [40] have produced a variant Cylindrical Algebraic Coverings (CAC) on CAD. Among other advantages, this might make discarding such x_0 *easier to implement, as* x_0 *would only be being considered* locally*, and "other reasons" would be irrelevant.*

Modern SAT solvers may be 10KLOC, and "hard to trust"[1], but Maple+CAD is probably 1MLOC, and much harder to trust if it says UNSAT, i.e. that the original problem has no solutions. We hope [41] that CAC may produce a proof outline for an instance that one could feed to a prover such as Coq or Isabelle, or quite possibly Lean: this is relevant as previous efforts to verify CAD algorithms in general have failed [42].

How might the ideas outlined in this paper actually all interface with a SAT solver to produce integrated SMT taking advantage of the strengths of both?

Acknowledgments

Thanks to many people, including my students and colleagues at Bath (especially Dr Uncu) and Coventry, and the members of EU H2020-FETOPEN-2016-2017-CSA project $\mathcal{SC}^2$ (712689). The writing was partially supported by EPSRC grant EP/T015713.

References

[1] G. Collins, Quantifier Elimination for Real Closed Fields by Cylindrical Algebraic Decomposition, in: Proceedings 2nd. GI Conference Automata Theory & Formal Languages, 1975, pp. 134–183.

[2] S. McCallum, A. Parusiński, L. Paunescu, Validity proof of Lazard's method for CAD construction, J. Symbolic Comp. 92 (2019) 52–69.

[3] J. Davenport, J. Heintz, Real Quantifier Elimination is Doubly Exponential, J. Symbolic Comp. 5 (1988) 29–35.

[4] C. Brown, J. Davenport, The Complexity of Quantifier Elimination and Cylindrical Algebraic Decomposition, in: C. Brown (Ed.), Proceedings ISSAC 2007, 2007, pp. 54–60.

[5] S. McCallum, An Improved Projection Operation for Cylindrical Algebraic Decomposition, Ph.D. thesis, University of Wisconsin-Madison Computer Science, 1984.

[6] D. Lazard, An Improved Projection Operator for Cylindrical Algebraic Decomposition, in: C. Bajaj (Ed.), Proceedings Algebraic Geometry and its Applications: Collections of Papers from Shreeram S. Abhyankar's 60th Birthday Conference, 1994, pp. 467–476.

[1] SAT contests now require them to produce an externally-verifiable proof of UNSAT.

[7] A. Nair, Curtains in Cylindrical Algebraic Decomposition, Ph.D. thesis, University of Bath, 2021. URL: https://researchportal.bath.ac.uk/en/studentTheses/curtains-in-cylindrical-algebraic-decomposition.

[8] J. Davenport, M. England, Need Polynomial Systems be Doubly-exponential?, in: G.-M. Greuel, T. Koch, P. Paule, A. Sommese (Eds.), International Congress on Mathematical Software ICMS 2016, volume 9725 of *Springer Lecture Notes in Computer Science*, 2016, pp. 157–164.

[9] M. England, J. Davenport, The Complexity of Cylindrical Algebraic Decomposition with Respect to Polynomial Degree, in: V. Gerdt, W. Koepf, W. Seiler, E. Vorozhtsov (Eds.), Proceedings CASC 2016, volume 9890 of *Springer Lecture Notes in Computer Science*, Springer, 2016, pp. 172–192. doi:10.1007/978-3-319-45641-6_12.

[10] J. von zur Gathen, T. Lücking, Subresultants revisited, Theoretical Computer Science 297 (2013) 199–239.

[11] S. McCallum, On Projection in CAD-Based Quantifier Elimination with Equational Constraints, in: S. Dooley (Ed.), Proceedings ISSAC '99, 1999, pp. 145–149.

[12] S. McCallum, On Propagation of Equational Constraints in CAD-Based Quantifier Elimination, in: B. Mourrain (Ed.), Proceedings ISSAC 2001, 2001, pp. 223–230.

[13] R. Bradford, J. Davenport, M. England, S. McCallum, D. Wilson, Truth table invariant cylindrical algebraic decomposition, J. Symbolic Comp. 76 (2016) 1–35.

[14] J. Davenport, M. England, The Potential and Challenges of CAD with Equational Constraints for SC-Square, in: Proceedings MACIS 2017: Mathematical Aspects of Computer and Information Sciences, 2017, pp. 280–285.

[15] M. England, R. Bradford, J. Davenport, Cylindrical Algebraic Decomposition with Equational Constraints, in: J. Davenport, M. England, A. Griggio, T. Sturm, C. Tinelli (Eds.), Symbolic Computation and Satisfiability Checking: special issue of Journal of Symbolic Computation, volume 100, 2020, pp. 38–71.

[16] C. Brown, S. McCallum, Enhancements to Lazard's Method for Cylindrical Algebraic Decomposition, in: F. Boulier, M. England, T. Sadykov, E. Vorozhtsov (Eds.), Computer Algebra in Scientific Computing CASC 2020, volume 12291 of *Springer Lecture Notes in Computer Science*, 2020, pp. 129–149. doi:https://doi.org/10.1007/978-3-030-60026-6_8.

[17] A. Nair, J. Davenport, G. Sankaran, On Benefits of Equality Constraints in Lex-Least Invariant CAD (Extended Abstract), in: SC-Square 2019: Satisfiability Checking and Symbolic Computation, volume 2460 of *CEUR WS Proceedings*, 2019, pp. 6:1–6:9. URL: http://ceur-ws.org/Vol-2460/paper6.pdf.

[18] A. Nair, J. Davenport, G. Sankaran, Curtains in CAD: Why Are They a Problem and How Do We Fix Them?, in: A. Bigatti, J. Carette, J. Davenport, M. Joswig, T. de Wolff (Eds.), Mathematical Software — ICMS 2020, volume 12097 of *Springer Lecture Notes in Computer Science*, Springer, 2020, pp. 17–26.

[19] V. Weispfenning, The Complexity of Linear Problems in Fields, J. Symbolic Comp. 5 (1988) 3–27.

[20] V. Weispfenning, Quantifier elimination for real algebra — the cubic case, in: Proceedings ISSAC 1994, 1994, pp. 258–263.

[21] V. Weispfenning, Quantifier elimination for real algebra — the quadratic case and beyond, AAECC 8 (1997) 85–101.

[22] M. Košta, New concepts for real quantifier elimination by virtual substitution, Ph.D. thesis, Universität des Saarlandes, 2016.
[23] K. Liiva, G. Passmore, P. Jackson, A note on real quantifier elimination by virtual term substitution of unbounded degree, https://homepages.inf.ed.ac.uk/pbj/papers/pas14.pdf, 2014.
[24] Z. Tonks, Poly-algorithmic Techniques in Real Quantifier Elimination, Ph.D. thesis, University of Bath, 2021. URL: https://researchportal.bath.ac.uk/en/studentTheses/poly-algorithmic-techniques-in-real-quantifier-elimination.
[25] Z. Tonks, A Poly-algorithmic Quantifier Elimination Package in Maple, in: J. Gerhard, I. Kotsireas (Eds.), Maple in Mathematics Education and Research 2019, volume 1125 of *Communications in Computer and Information Science*, 2020, pp. 171–186.
[26] E. Ábrahám, Building Bridges between Symbolic Computation and Satisfiability Checking, in: D. Robertz (Ed.), Proceedings ISSAC 2015, 2015, pp. 1–6.
[27] V. Weispfenning, A New Approach to Quantifier Elimination for Real Algebra, in: B. Caviness, J. Johnson (Eds.), Quantifier Elimination and Cylindrical Algebraic Decomposition, Springer-Verlag, 1998, pp. 376–392.
[28] R. Fukasaku, H. Iwane, Y. Sato, Real Quantifier Elimination by Computation of Comprehensive Gröbner Systems, in: D. Robertz (Ed.), Proceedings ISSAC 2015, 2015, pp. 173–180.
[29] P. Pedersen, M.-F. Roy, A. Szpirglas, Counting Real Zeroes in the Multivariate Case, in: Proceedings MEGA ’92, 1993, pp. 203–224.
[30] V. Weispfenning, Comprehensive Gröbner Bases, J. Symbolic Comp. 14 (1992) 1–29.
[31] P. Aubry, D. Lazard, M. Moreno Maza, On the Theories of Triangular Sets, J. Symbolic Comp. 28 (1999) 105–124.
[32] G. Gallo, B. Mishra, Efficient Algorithms and Bounds for Wu-Ritt Characteristic Sets, in: T. Mora, C. Traverso (Eds.), Proceedings MEGA 1990, 1991, pp. 119–142.
[33] A. Dickenstein, N. Fitchas, M. Giusti, C. Sessa, The Membership Problem for Unmixed Polynomial Ideals is Solvable in Single Exponential Time, Discrete Appl. Math. 33 (1991) 73–94.
[34] C. Chen, M. Moreno Maza, B. Xia, L. Yang, Computing Cylindrical Algebraic Decomposition via Triangular Decomposition, in: J. May (Ed.), Proceedings ISSAC 2009, 2009, pp. 95–102.
[35] C. Chen, M. Moreno Maza, Quantifier Elimination by Cylindrical Algebraic Decomposition Based on Regular Chains, in: K. Nabeshima (Ed.), Proceedings ISSAC 2014, 2014, pp. 91–98.
[36] C. Chen, M. Moreno Maza, Cylindrical Algebraic Decomposition in the RegularChains Library, in: Proceedings Mathematical Software — ICMS 2014, 2014, pp. 425–433.
[37] C. Chen, M. Moreno Maza, An Incremental Algorithm for Computing Cylindrical Algebraic Decompositions, in: R. Feng, W.-s. Lee, Y. Sato (Eds.), Computer Mathematics, Springer Berlin Heidelberg, 2014, pp. 199–221. URL: http://dx.doi.org/10.1007/978-3-662-43799-5_17. doi:10.1007/978-3-662-43799-5_17.
[38] R. Bradford, C. Chen, J. Davenport, M. England, M. Moreno Maza, D. Wilson, Truth Table Invariant Cylindrical Algebraic Decomposition by Regular Chains, in: Proceedings CASC 2014, 2014, pp. 44–58.
[39] R. Bradford, J. Davenport, M. England, A. Sadeghimanesh, A. Uncu, The DEWCAD Project: Pushing Back the Doubly Exponential Wall of Cylindrical Algebraic Decomposition, ACM

Comm. Computer Algebra 55 (2021) 107–111. URL: https://arxiv.org/abs/2106.08740.
[40] E. Ábrahám, J. Davenport, M. England, G. Kremer, Deciding the Consistency of Non-Linear Real Arithmetic Constraints with a Conflict Driven Search Using Cylindrical Algebraic Coverings, Journal of Logical and Algebraic Methods in Programming Article 100633 119 (2021).
[41] E. Ábrahám, J. Davenport, M. England, G. Kremer, Z. Tonks, New Opportunities for the Formal Proof of Computational Real Geometry?, SC^2'20: Fifth International Workshop on Satisfiability Checking and Symbolic Computation CEUR Workshop Proceedings 2752 (2020) 178–188.
[42] C. Cohen, A. Mahboubi, Formal Proofs in Real Algebraic Geometry: From Ordered Fields to Quantifier Elimination, Logical Methods in Computer Science 8 (2012) 1–40.
[43] J. Davenport, Computer Algebra for Cylindrical Algebraic Decomposition, Technical Report TRITA-NA-8511 NADA KTH Stockholm (Reissued as Bath Computer Science Technical Report 88-10), 1985. URL: http://staff.bath.ac.uk/masjhd/TRITA.pdf.

A. Collins Complexity

In [1], he derived $e_d = 2n + 8$, whereas Table 1 has $(\log_2 3)n + O(1)$. The more precise figure comes from three improvements.

1. [1, Theorem 15] is derived from [1, Theorem 14] "by observing that $3^h \leq 2^{2h}$", so we should use Theorem 14 directly.
2. "For example, the analysis depends strongly on the root separation theorem, and it seems likely that this theorem is far from optimal" [1, p. 173]. [43, Proposition 8] provides a better theorem, in that it shows that $C(f)$, the number of subdivisions needed to separate all the roots of a polynomial f of degree d, is asymptotically no worse than that needed to separate the closest pair.
3. If the $\alpha_i^{(j)}$ are the real roots of the polynomials f_i, and we need to separate all the $\alpha_i^{(j)}$, [1] considers this as $C(\prod_i f_i)$. But it is $\sum_{i,j} C(f_i f_j)$, which is smaller because of [43, Proposition 8].