Cylindrical Algebraic Decomposition with Equational Constraints

Matthew England[a], Russell Bradford[b], James H. Davenport[b]

[a]*Faculty of Engineering, Environment and Computing, Coventry University, UK*
[b]*Faculty of Science, University of Bath, UK*

Abstract

Cylindrical Algebraic Decomposition (CAD) has long been one of the most important algorithms within Symbolic Computation, as a tool to perform quantifier elimination in first order logic over the reals. More recently it is finding prominence in the Satisfiability Checking community as a tool to identify satisfying solutions of problems in nonlinear real arithmetic.

The original algorithm produces decompositions according to the signs of polynomials, when what is usually required is a decomposition according to the truth of a formula containing those polynomials. One approach to achieve that coarser (but hopefully cheaper) decomposition is to reduce the polynomials identified in the CAD to reflect a logical structure which reduces the solution space dimension: the presence of Equational Constraints (ECs).

This paper may act as a tutorial for the use of CAD with ECs: we describe all necessary background and the current state of the art. In particular, we present recent work on how McCallum's theory of reduced projection may be leveraged to make further savings in the lifting phase: both to the polynomials we lift with and the cells lifted over. We give a new complexity analysis to demonstrate that the double exponent in the worst case complexity bound for CAD reduces in line with the number of ECs. We show that the reduction can apply to both the number of polynomials produced and their degree.

Keywords: cylindrical algebraic decomposition, non linear real arithmetic

Email addresses: Matthew.England@coventry.ac.uk (Matthew England), R.J.Bradford@bath.ac.uk (Russell Bradford), J.H.Davenport@bath.ac.uk (James H. Davenport)

1. Introduction

1.1. Cylindrical algebraic decomposition

A Cylindrical Algebraic Decomposition (CAD) splits $\mathbb{R}^n$ into cells to maintain an invariance structure relative to an input. Traditionally, the cells are produced to be *sign-invariant* for a set of input polynomials: meaning that throughout each cell, each of those polynomials has a constant sign. The first CAD algorithm was introduced by Collins (1975) to perform Quantifier Elimination (QE) over real closed fields. We describe the necessary details and terminology for CAD in Section 2.1. The invariance property of a CAD means that problems for non-linear polynomial systems such as QE are reduced to testing a finite number of sample points; with the nature of the cells produced allowing for the easy generation of solution descriptions.

However, the use of CAD is often limited by the complexity of computing one. CAD is known to have worst case complexity doubly exponential in the number of variables (Davenport and Heintz, 1988; Brown and Davenport, 2007). Broadly speaking (see Theorem 5 for a precise result) if the input has m polynomials of degree at most d then CAD complexity could be in the order of $(2dm)^{2^{O(n)}}$. For some problems there exist algorithms with better complexity (see the textbook by Basu et al. (2006) for example), and there are also many specialised algorithms for restricted inputs; but CAD implementations remain the only general purpose approach for many problems.

This complexity statement is obtained by considering the necessary size of the output for certain examples, and so can only be reduced with changes to the output requirements. Further, unlike some other theoretical results, this one is clearly felt in practice: when increasing dimensionality one will hit the "doubly exponential wall" where progress becomes infeasible. However, extensive investigation into CAD and its sub-algorithms has allowed for the wall to be "pushed back" to the point of allowing many useful computations.

Applications of CAD include: motion planning (Schwartz and Sharir, 1983), weight minimisation for truss design (Charalampakis and Chatzigiannelis, 2018), epidemic modelling (Brown et al., 2006), steady state analysis of biological networks (Bradford et al., 2017), economic reasoning (Mulligan et al., 2018a), artificial intelligence to pass exams (Wada et al., 2016), parametric optimisation (Fotiou et al., 2005), theorem proving (Paulson, 2012), derivation of optimal numerical schemes (Erascu and Hong, 2016), reasoning with multi-valued functions (Davenport et al., 2012), and much more.

1.2. CAD and SC²

CAD has long been important within Symbolic Computation with implementations in multiple computer algebra systems. However, in recent years CAD has been of interest to the separate community of Satisfiability Checking. There, search based algorithms developed for the Boolean SAT problem, make use of heuristics and learning (see the textbook by Biere et al. (2009) for details). Success here led to research on domains other than the Booleans, and Satisfiability Module Theory (SMT)-Solvers which use SAT algorithms on the Boolean skeleton of a problem with queries to theory solvers to see if a satisfying Boolean assignment is valid in the domain (learning new clauses if not) (Barrett et al., 2009; Kroening and Strichman, 2013).

For the SMT domain of non-linear real arithmetic (NRA), CAD and more generally computer algebra systems can play the role of such theory solvers[1]. SMT-RAT contains a tailored CAD implementation for use in SMT (Loup et al., 2013; Kremer and Ábrahám, 2019), while Z3 contains an algorithm by Jovanovic and de Moura (2012) which uses CAD theory without producing actual CADs. The latter inspired new developments in symbolic computation such as non-uniform CAD by Brown (2015). Further collaboration is informed by the SC² project: forging interaction between **S**ymbolic **C**omputation and **S**atisfiability **C**hecking (Ábrahám et al., 2016).

CADs are produced relative to a problem statement expressed in logic connectives between atoms involving (potentially non-linear) polynomials with integer coefficients. The original CAD algorithm produces decompositions according to the signs of these polynomials, essentially ignoring the logical structure entirely and so producing decompositions fine enough to solve all problems for all logical formulae formed by those polynomials. A Satisfiability Checking approach like that of Jovanovic and de Moura (2012) takes an opposite focus, analysing and extending the logical skeleton of the formula until a solution is found with the correct algebraic properties. In order to derive a full solution from a CAD what is truly required is a decomposition on whose cells the truth of the overall logical formula is constant[2].

[1] However, as discussed by Ábrahám et al. (2016) a more custom approach is beneficial.

[2] A sign-invariant decomposition for the polynomials in the formula achieves this, but with far more cells and computation than required.

1.3. Equational constraints

A CAD complexity analysis, such as that in Section 6, does not just conclude the upper bound $(2dm)^{2^{O(n)}}$: it actually shows that in one dimension we must isolate the roots of at most M polynomials of degree D, where $D = d^{2^{O(n)}}$ and $M = m^{2^{O(n)}}$. The same orders have been found for lower bounds: for D by Davenport and Heintz (1988) and for M by Brown and Davenport (2007). The formulae demonstrating this are not straightforward but the underlying polynomials *are* surprisingly simple (all bar two linear with each only involving a bounded number of variables, independent of n). This demonstrates that the difficulty of CAD resides in the complicated number of ways simple polynomials can interact. Improvement must come from reducing the number of interactions we track.

Definition 1. *A* Quantifier Free Tarski Formula (QFF) *is made up of a finite number of atoms connected by the standard Boolean operators* $\wedge$, $\vee$ *and* $\neg$. *The atoms are statements about the signs of polynomials with integer coefficients:* $f \, \sigma \, 0$ *where* $\sigma \in \{=, <, >\}$ *(and by combination also* $\{\leq, \geq, \neq\}$*).*

Definition 2. *An* Equational Constraint *(EC) is a polynomial equation logically implied by a QFF. If it is an atom of the formula it is said to be* explicit *and if not then it is* implicit.

Example 1. *Let* f *and* g *be polynomials. (a) The formula* $f = 0 \wedge g > 0$ *has explicit EC* $f = 0$. *(b) The formula* $f = 0 \vee g = 0$ *has no explicit EC but it does have the implicit EC* $fg = 0$. *(c) The formulae* $f^2 + g^2 \leq 0$ *also has no explicit EC but this one has two implicit ECs* $f = 0$ *and* $g = 0$.

Collins (1998) was the first to suggest that CAD could be simplified in the presence of an EC. He noted that a CAD need only be sign-invariant for the defining polynomial of an EC, and sign-invariant for any others only within those cells where the EC polynomial is zero. He sketched an intuitive approach to produce this by refining the polynomials identified by his CAD algorithm. This approach was formalised and verified by McCallum (1999b). A complexity analysis (Bradford et al., 2016, Section 2) showed that making use of a single EC in this way reduces the double exponent of m in the complexity bound for CAD by 1. Some natural questions arising are:

- Can savings be made iteratively in the presence of multiple ECs?
- Do those savings further reduce the double exponent?
- Can corresponding savings be made for the double exponent of d?

The first question was answered affirmatively by McCallum (2001), although the extension was not trivial[3]. The other questions are answered affirmatively by the present paper. Such questions are of growing importance as CAD finds new application domains with increasing numbers of equations: e.g. in biology by Bradford et al. (2017) and England et al. (2017); and in economics by Mulligan et al. (2018a,b). Indeed, many problems that arise in the Satisfiability Checking context contain far more equalities than inequalities (see the NRA benchmarks in the SMT-LIB (Barrett et al., 2016)).

1.4. Contribution and plan

In Section 2 we define the necessary CAD terminology and revise the theory for projection, and reduced projection in the presence of an EC, of McCallum (1998, 1999b, 2001). Then in Section 3 we present recent work on how to leverage this for savings elsewhere in CAD. In Section 4 we propose and verify the corresponding algorithm.

We demonstrate the benefit first with a worked example in Section 5 and then a complexity analysis in Section 6. The latter observes the double exponent in the bound on the number of projection polynomials reducing by the number of ECs. In Section 7 and 8 we explain how a similar reduction can be observed for their degree if we assume CAD input is pre-processed with a Gröbner Basis: a common step in CAD implementations but this is the first theoretical justification for it. Together these show that CAD is doubly exponential in number of variables minus number of ECs.

In Section 9 we examine the main caveat: an assumption of primitivity to the ECs. We demonstrate its presence in the key results of Davenport and Heintz (1988); Brown and Davenport (2007) proving worst case CAD complexity, and discuss what might be done. We finish by discussing lessons for the Satisfiability Checking community who may call CAD within SMT-solvers in Section 10, and giving a summary in Section 11.

The main contributions in this paper were presented at ISSAC 2015 (England et al., 2015) and CASC 2016 (England and Davenport, 2016). These conference publications addressed the savings in the number of polynomials and their degrees separately. The present paper unifies the results into a coherent whole providing a single statement of the state of the art. It also expands on some details, such as the need for primitivity in Section 9.

[3]and contained a small mistake as described in Remark 3.

We include all necessary background theory, allowing the paper to act as a tutorial for CAD with ECs, timely given the increased use of CAD outside of computer algebra systems, as part of satisfiability checkers. We further expose the results to the wider SC^2 by considering implications for CAD in SMT-solvers in Section 10.

2. Background Material

2.1. CAD computation and terminology

We work under variable ordering $\boldsymbol{x} = x_1 \prec \ldots \prec x_n$. The *main variable* of a polynomial or formula (mvar) is the greatest present under the ordering.

Definition 3. *A* Cylindrical Algebraic Decomposition (CAD) *is a decomposition of* $\mathbb{R}^n$ *into connected cells such that:*

- *each cell is a* semi-algebraic set *meaning it is defined by a finite sequence of polynomial equations or inequalities; and*
- *the cells are arranged* cylindrically *meaning the projections of any two cells in the decomposition on any lower dimensional space with respect to the ordering are either equal of disjoint.*

The latter condition means that each CAD cell is defined by a sequence of conditions: $c_1(x_1), c_2(x_1, x_2), \ldots, c_n(x_1, \ldots, x_n)$ where each c_i is one of:

$$\ell_i(x_1, \ldots, x_{i-1}) < x_i \tag{1}$$
$$\ell_i(x_1, \ldots, x_{i-1}) < x_i < u_i(x_1, \ldots, x_{i-1}) \tag{2}$$
$$x_i < u_i(x_1, \ldots, x_{i-1}) \tag{3}$$
$$x_i = s_i(x_1, \ldots, x_{i-1}) \tag{4}$$

The ℓ_i, u_i, s_i are constants when $i = 1$ and otherwise most likely an indexed root expression (a particular root of a polynomial). The former condition tells us that an equivalent semi-algebraic description can be found.

We describe the computation scheme and terminology that the Collins-descended CAD algorithms share. Assume a set of input polynomials (possibly derived from formulae). The first phase of CAD, *projection*, applies projection operators repeatedly, each time producing another set of polynomials in one less variable (following the variable ordering). Together these are the *projection polynomials* used in the second phase, *lifting*.

First $\mathbb{R}$ is decomposed into cells according to the real roots of polynomials univariate in x_1. Each cell is either a point and therefore its own sample, or an interval inside which we choose a convenient sample (often the dyadic rational with least denominator). We next decompose $\mathbb{R}^2$ by repeating a process over each cell in $\mathbb{R}^1$. In each case we take the bivariate projection polynomials in (x_1, x_2), evaluate them at the sample point of the cell in $\mathbb{R}^1$ to give univariate polynomials in x_2 whose roots we can count and isolate.

The cells identified in $\mathbb{R}^2$ fall into two categories. *Sections* are defined according to the vanishing of a polynomial as in (4), and correspond to the real root of a univariate polynomial. *Sectors* are usually defined as the regions between two sections as in (2), corresponding to the intervals between real roots of a univariate polynomial. The exceptions are the two infinite sections at either end of the decomposition as in (3) and (1); or if there was no need to decompose at all there may be a single infinite sector with no restrictions on x_2. In each case the sample point is extended from that of the underlying cell: for sections to include the algebraic number isolated for the real root and for sectors to include any convenient number from the interval (certainly in $\mathbb{Q}$). Together the sections and sectors form a *stack* over the cell in $\mathbb{R}^1$.

Taking the union of these stacks gives the CAD of $\mathbb{R}^2$. The process may then be repeated, each time producing a CAD of larger $\mathbb{R}^i$, until a CAD of $\mathbb{R}^n$ is produced. The subspaces $\mathbb{R}^i$ decomposed are those implied by the ordering: (x_1)- space, (x_1, x_2)-space, (x_1, x_2, x_3)-space etc.

Cells are represented at a minimum with a sample point and *index*. The latter is a list of integers, with the kth describing variable x_k according to the ordered real roots of the projection polynomials. If the integer is $2i$ the cell is a section corresponds to the ith root (counting low to high) and if $2i + 1$ it is the adjacent sector[4]. The projection operator is chosen so polynomials are *delineable* in a cell: the portion of their zero set in the cell consists of disjoint sections. A set of polynomials are *delineable* if each is delineable individually, and the sections of different polynomials are identical or disjoint. If all projection polynomials are delineable then the input polynomials must be *sign-invariant*: have constant sign in each cell of the CAD.

There have been a great many extensions and optimisations of CAD since its inception, with the survey article by Collins (1998) highlighting those from the first 20 years. Since then further highlights have included:

[4]The dimension of a cell is hence easily identified from the index.

symbolic-numeric lifting schemes (Strzeboński, 2006; Iwane et al., 2009); local projection approaches (Brown, 2013; Strzeboński, 2016); comprehensive Gröbner basis approaches (Fukasaku et al., 2015) and decompositions via complex space (Chen et al., 2009; Bradford et al., 2014).

2.2. Projection operators

A key improvement to CAD has been in the projection operator to reduce the number of projection polynomials computed (Hong, 1990; McCallum, 1998, 1999b, 2001; Brown, 2001; Bradford et al., 2013a; Han et al., 2014; Bradford et al., 2016).

The minimal complete projection operator (for sign-invariant CAD) proposed is that of Lazard (1994), however the proof of its correctness was shown to be flawed by McCallum and Hong (2016). Shortly before this article went to press a corrected proof was published by McCallum et al. (2019) (requiring changes elsewhere in the CAD lifting phase). The theory in the present paper uses the family of projection operators by McCallum (1998, 1999b, 2001) but, in due course they may be improved by extending Lazard's family into the EC theory. We note that the relative savings from the ECs laid out in the present paper would still be maintained if recast into Lazard projection.

Throughout, let cont, prim, disc, ldcf and coeff denote the content, primitive part, discriminant, leading coefficient, and set of all coefficients of polynomials respectively (in each case taken with respect to a given mvar). When applied to a set of polynomials we interpret these as applying the operation to each polynomial in the set. e.g.

$$\begin{aligned} \operatorname{cont}(A) &= \{\operatorname{cont}(f) \,|\, f \in A\}, \\ \operatorname{coeff}(A) &= \cup_{f \in A} \operatorname{coeff}(A). \end{aligned}$$

We let res denote the resultant of a pair of polynomials, and for a set

$$\operatorname{res}(A) = \{\operatorname{res}(f_i, f_j) \,|\, f_i \in A, f_j \in A, f_i \neq f_j\}.$$

Recall that a set $B \subset \mathbb{Z}[\mathbf{x}]$ is an *irreducible basis* if the elements of B are of positive degree in the mvar, irreducible and pairwise relatively prime. Throughout this section suppose B is an irreducible basis for a set of polynomials, and further that every element of B has mvar x_n and that $F \subseteq B$.

We may now define the projection operators introduced respectively by McCallum (1998, 1999b, 2001):

$$P(B) := \text{res}(B) \cup \text{disc}(B) \cup \text{coeff}(B), \tag{5}$$

$$P_F(B) := P(F) \cup \{\text{res}(f, g) \mid f \in F, g \in B \setminus F\}, \tag{6}$$

$$P_F^*(B) := P_F(B) \cup \text{disc}(B \setminus F) \cup \text{coeff}(B \setminus F). \tag{7}$$

In the general case with A a set of polynomials and $E \subseteq A$ we proceed with projection by: letting B and F be irreducible bases of the primitive parts of A and E respectively; applying the operators as defined above; and then taking the union of the output with $\text{cont}(A)$.

Remark 1. *We see that (6) is contained in (5) and will usually be smaller. It excludes discriminants and coefficients of $B \setminus F$ which are then reinstated by (7). It also excludes those resultants which involve two polynomials from $B \setminus F$, an exclusion that is maintained by (7). Thus we have $(6) \subseteq (7) \subseteq (5)$.*

Remark 2. *The full set of coefficients of a polynomial is usually unnecessary for CAD (Brown, 2001). We require knowledge of when the polynomial's degree drops, and thus the vanishing of the leading coefficient is of most importance. But in the case that it did vanish then the next coefficient becomes leading and thus must also be studied. To guarantee correctness in all cases the full set is taken but most implementations will safely optimise this. E.g. if the leading coefficient is constant then it can never vanish and no further coefficients need to considered.*

Remark 3. *Operator (7) is different to the $P_F^*(B)$ of McCallum (2001), which excluded the coefficients of $B \setminus F$. The 2001 definition was a mistake, pointed out to us by the anonymous referee of this paper and confirmed in a private communication with McCallum. However, it is not necessary for us to prove a corrected version of the theorems from that paper because, as was noted by McCallum (2001) (just after Theorem 2.1) if we allow the additional coefficients then we can assume degree invariance and thus use the existing theorems of McCallum (1998) to validate $P_F^*(B)$.*

2.3. Sign invariant CAD Projection

The theorems below use the idea of *order-invariance*, meaning each polynomial has constant order of vanishing within each cell, which of course implies sign-invariance. We say that a polynomial with mvar x_k is *nullified* over a cell in $\mathbb{R}^{k-1}$ if it vanishes identically throughout.

Theorem 1 (McCallum (1998), Thm. 1)**.** *Let S be a connected submanifold of $\mathbb{R}^{n-1}$ in which each element of $P(B)$ is order-invariant. Then on S, each element of B is either nullified or analytic delineable*[5]*. Further, the sections of elements of B that are not nullified are either identical or pairwise disjoint, and each element of B is order-invariant on such sections.*

Suppose we apply P repeatedly to generate projection polynomials. Repeated use of Theorem 1 concludes that a CAD produced by lifting with respect to these projection polynomials is order-invariant so long as no projection polynomial with mvar x_k is nullified over a cell in the CAD of $\mathbb{R}^{k-1}$. This condition is known as *well-orientedness*. It is common for problems to be well oriented and the condition can be easily checked for during lifting[6]. In the case that a problem is not well-oriented we cannot conclude sign-invariance (at least over the cell in which nullification occurred). There are some cases where we can rescue the computation (see the report by Brown (2005)) but in some cases the only option will be to use an alternative complete projection operator, such as that of Hong (1990). We note that the Lazard operator, recently validated by McCallum et al. (2019), removes the well-orientedness condition for sign-invariant CAD[7].

2.4. CAD projection for a formula with a single EC

A second theorem allows us to understand how $P_E(A)$ is validated.

Theorem 2 (McCallum (1999b), Thm. 2.2)**.** *Let f and g be integral polynomials with mvar x_n and $r(x_1, \ldots, x_{n-1})$ be their resultant. Suppose $r \neq 0$. Let S be a connected subset of $\mathbb{R}^{n-1}$ on which f is delineable and r order-invariant. Then g is sign-invariant in every section of f over S.*

Suppose A was derived from a formula with EC defined by $E = \{f\}$, and that we apply $P_E(A)$ once and then P repeatedly to generate projection polynomials. Assuming the input is well-oriented, we can use Theorem 1 to conclude the CAD of $\mathbb{R}^{n-1}$ order invariant for $P_E(A)$. The CAD of $\mathbb{R}^n$ is then sign-invariant for E using Theorem 1 and sign-invariant for A in the sections of E using Theorem 2. Hence the CAD is truth-invariant for the formula.

[5] A variant on delineability defined by McCallum (1998).

[6] Recall that to lift over a cell we first substitute the cell sample point into the polynomials with main variable one higher: so at this stage we check if any vanish entirely.

[7] Instead a modified lifting stage checks for nullification and adapts such polynomials to recover the lost information.

2.5. CAD projection with multiple ECs

We now consider the case of multiple ECs. We could of course apply the previous technology by designating one EC for special use and treating the rest as any other constraint (heuristics can help with the choice (Bradford et al., 2013b)). But this does not gain any further advantage from the additional ECs, which should be reducing the dimension of our solution space. It is important to note that we cannot simply add multiple EC defining polynomials into E and use $P_E(A)$. That would result in a CAD truth-invariant for the disjunction of the ECs, not the conjunction implied by multiple ECs.

Let us first assume that we have two ECs: one whose mvar is that of the system, x_n, and another whose mvar is x_{n-1}. Consider applying first the operator $P_E(A)$ where E defines the first EC and then $P_{E'}(A')$ where $A' = P_E(A)$ and $E' \subseteq A'$ contains the second EC. Unfortunately, Theorem 2 does not validate this approach. While it could be applied once for the CAD of $\mathbb{R}^{n-1}$ it cannot then validate the CAD of $\mathbb{R}^n$ because the first application of the theorem provided sign-invariance while the second requires the stronger condition of order invariance. The approach is acceptable if $n = 3$ (since in two variables the conditions are equivalent for squarefree bases).

Example 2. *We consider the formula* $\phi = f_1 = 0 \wedge f_2 = 0 \wedge g \geq 0$ *with the following polynomials:*

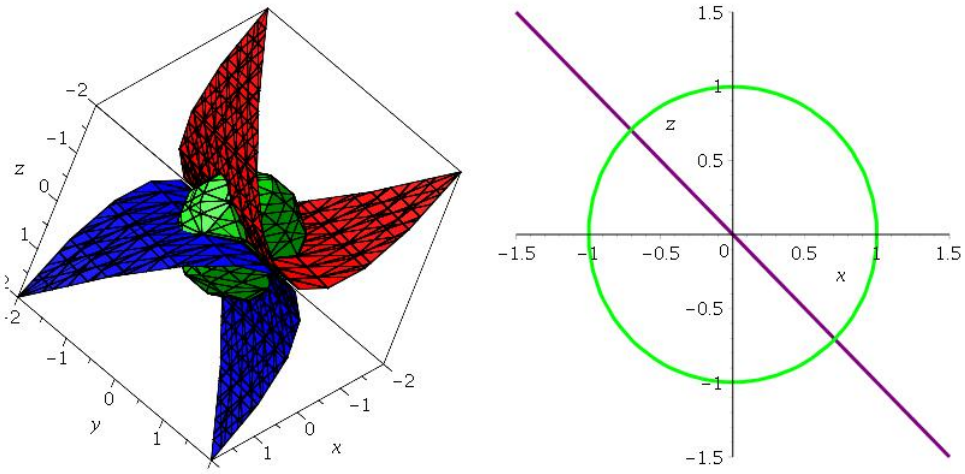
$$f_1 = x + y^2 + z, \quad f_2 = x - y^2 + z, \quad g = x^2 + y^2 + z^2 - 1.$$

The polynomials are graphed in Figure 1 where g *is the sphere,* f_1 *the upper surface and* f_2 *the lower. We see that* f_1 *and* f_2 *only meet when* $y = 0$ *and this projection is on the right of Figure 1. It shows that the solution requires* $|x| \geq \sqrt{2}/2$ *and* $z = -x$*. How could this be ascertained using CAD?*

With variable ordering $z \succ y \succ x$ *a sign-invariant CAD for* (f_1, f_2, g) *has 1487 cells using the* QEPCAD-B *program by Brown (2003). We could then test a sample point of each cell to identify the ones where* ϕ *is true. It is preferable to use the presence of ECs. Declaring an EC to* QEPCAD *will ensure it uses the algorithm by McCallum (1999b) based on a single use of* $P_E(A)$ *followed by* P*. Either choice results in 289 cells. In particular, the solution set is described using 8 cells: all have* $y = 0, z = -x$ *but the* x*-coordinate unnecessarily splits cells at* $\frac{1}{2}(1 \pm \sqrt{6})$*. This is identified due to the projection polynomial* $d = \text{disc}_y(\text{res}_z(f_i, g))$*.*

For problems with $n > 3$ it is still possible to make use of multiple ECs. However, we must include the extra information necessary to provide

Figure 1: The polynomials from Example 2.



order-invariance of the non-EC polynomials in the sections of ECs. The following theorem explains that discriminants are required to maintain order invariance, along with degree invariance (and hence coefficients).

Theorem 3 (McCallum (1998), Thm. 2)**.** *Let f be a polynomial with main variable x_n with positive degree and $d = \mathrm{disc}(f)$ which we suppose to be non-zero. Let S be a connected submanifold of $\mathbb{R}^{n-1}$ on which f is degree invariant, and does not vanish identically, and in which d is order invariant. Then f is analytic delineable on S and is order-invariant in each section of f over S.*

Theorem 3 was used originally as a tool to prove Theorem 1: it gives us the order invariance of polynomials and individual delineability (adding the resultants then extends this to delineability of the set). We can use it again now to show that the strengthening of (6) with the added discriminants and coefficients to form (7) allows for the order invariance conclusion needed for continued application and validation of the operator.

Suppose we have a formula with two ECs, one with mvar x_n and the other with mvar x_{n-1}. We may now use a reduced operator twice. We first calculate $A' = P_E(A)$ where E contains the defining polynomial of the first EC, and then $P^*_{E'}(A')$ where E' contains the defining polynomial of the other. Subsequent projections simply use P. When lifting we use: first Theorem 1

to verify the CAD of $\mathbb{R}^{n-2}$ as order-invariant for $P^*_{E'}(A')$; then Theorem 1 to verify the CAD of $\mathbb{R}^{n-1}$ delineable and order-invariant for E', and Theorem 3 to verify it order-invariant and delineable for the individual polynomials of A' in the sections of E'; and finally Theorem 1 and 2 to verify the CAD of $\mathbb{R}^n$ order-invariant for E and sign-invariant for A in those cells that are both sections of E and E'.

2.6. EC propagation

We can now maximise savings in projection when we have ECs in different main variables. Further, if we have two ECs with the same mvar we can usually derive another with a lower mvar by taking the resultant of the two defining polynomials. McCallum (2001) defined this as *EC propagation*. The process requires the two original ECs to be independent, i.e. the satisfaction of one does not imply the satisfaction of the other.

Given additional ECs one can perform multiple rounds of propagation to obtain implicit ECs in a sequence of different main variables. Actually in this case there will be more ECs than we are able to use. For example, given (independent) ECs $f_i(x, y, z) = 0$ for $i = 1, 2, 3$ in variables $z \prec y \prec x$ then a further three implicit ECs can be found with main variable y and another three with main variable x:

$$r_1 = \text{res}_z(f_1, f_2), \quad r_2 = \text{res}_z(f_1, f_3), \quad r_3 = \text{res}_z(f_2, f_3),$$
$$R_1 = \text{res}_y(r_1, r_2), \quad R_2 = \text{res}_y(r_1, r_3), \quad R_3 = \text{res}_y(r_2, r_3).$$

Of course, the latter three will not be independent (the vanishing of one should imply the vanishing of another) but even then there may still be questions of efficiency over which to use. While Bradford et al. (2013b)h have developed heuristics to help with the choice of which EC to use, there is likely room for improvement[8].

Example 3. *Consider again the example problem from Example 2. We can propagate the two ECs $f_1 = 0$ and $f_2 = 0$ to find implicit EC $r_1 = 0$ as defined above. The resultant of the two defining polynomials is $-2y^2$ so we may simplify the EC to $y = 0$.*

[8]One possibility is the use of machine learning classifiers to make such choices. This is a growing topic within mathematical software, with a recent survey given by England (2018). It has been applied to CAD by Huang et al. (2014, 2016).

If we declare both ECs in z to QEPCAD *then it will perform the propagation for us and use reduced projection twice. It will actually apply $P_E(A)$ twice (allowed since $n = 3$) to produce a CAD with 133 cells. The solution set is now described using only 4 cells (the minimum possible). Note that d (see Example 2) was no longer produced as a projection polynomial.*

3. Reductions in the Lifting Stage

The first main contribution of the present paper is to realise that the theorems from the previous section allow for significant savings in the lifting phase (beyond those achieved from reduced projection). To implement these we must discard two embedded principles of CAD:

- That the projection polynomials are a fixed set.
- That the invariance structure of the final CAD can be expressed in terms of sign-invariance of polynomials.

The first was also abandoned by Chen et al. (2009); Bradford et al. (2014); Jovanovic and de Moura (2012); Brown (2015), while the second by Brown and McCallum (2005); McCallum and Brown (2009).

3.1. Minimising the number of polynomials used for lifting

Consider Theorem 2: it allows us to conclude that g is sign-invariant in the sections of f produced over a CAD of $\mathbb{R}^{n-1}$ order-invariant for $P_{\{f\}}(\{f, g\})$. Therefore, it is sufficient to perform the final lift with respect to f only (decompose cylinders according to the roots of f but not g). The decomposition imposes sign-invariance for f while Theorem 2 guarantees it for g in the cells where it matters (where those signs could change the truth of the formula).

Example 4. *We return to Example 2. Recall that designating either EC and using the algorithm by McCallum (1999b) produced a CAD with 289 cells. If we follow this approach but lift only with respect to the designated EC at the final step (implemented in the* MAPLE *package by England et al. (2014b)) we obtain a CAD with 141 cells: less than half the output.*

This improved lifting follows from the theorems of McCallum (1999b), but was only noticed 15 years later during the generalisation of that work

to the case of multiple formulae by Bradford et al. (2013a, 2016). Experiments there demonstrated its importance, particularly for problems with many constraints: see Section 8.3 of (Bradford et al., 2016).

When we apply a reduced operator at two levels then we can make such reductions at both the corresponding lifts.

Example 5. *We return to the problem from Example 2. Set* $A = \{f_1, f_2, g\}$ *and* $E = \{f_1\}$*. The first projection to eliminate* z *finds*

$$P_E(A) = \{y, y^4 + 2xy^2 + 2x^2 + y^2 - 1\}.$$

These are the resultants of f_1 *with* f_2 *and* g *(after the first is simplified as discussed in Example 3). The discriminant of* f_1 *was a constant and so could be discarded, as was its leading coefficient (meaning no further coefficients were required as explained in Remark 2). Set* $A' = P_E(A)$ *and* $E' = y$ *(since* y *defines an EC for the problem as discussed in Example 3). We have* $P_{E'}(A') = \{R\}$ *where*

$$R = \text{res}_y(y, y^4 + 2xy^2 + 2x^2 + y^2 - 1) = 2x^2 - 1.$$

The other possible entries (the discriminants and coefficients from E'*) are all constants. We hence build a 5 cell CAD of the real line with respect to the two real roots of* R*. We then lift above each cell with respect to* y *only, in each case splitting the cylinder into three cells about* $y = 0$*, to give a CAD of* $\mathbb{R}^2$ *with 15 cells.*

Finally, we lift over each of these 15 cells with respect to f_1 *to give 45 cells of* $\mathbb{R}^3$*. This compares to 133 from* QEPCAD*, which used reduced projection but then lifted with* all *projection polynomials. No polynomials were nullified, so using Theorems 1 and 2, the output is concluded truth-invariant for* ϕ*.*

Remark 4. *We note that not only is the additional lifting that* QEPCAD *performed unnecessary for the problem at hand, it also provides no further structure on the output. For example, if we had lifted with respect to* f_2 *at the final stage in Example 5 then we would be doing so without the knowledge that it is delineable. Hence splitting the cylinder at the sample point offers no guarantee that the cells produced are sign-invariant away from that sample point. So the extra work does not allow us to conclude that* f_2 *is sign-invariant (except on sections of* f_1 *where we could have concluded it already).*

Remark 5. *The improvement outlined above not only decreases output size (and computation time) but also the risk of failure*[9] *from non well-oriented input: we only need worry about nullification of polynomials we lift with.*

3.2. Minimising the cells for stack generation

We can achieve more savings by abandoning the aim of producing a CAD sign-invariant with respect to any polynomial, instead insisting only on truth-invariance. We may then lift cells already known to be false trivially to cylinders. The idea of avoiding computations over false cells was presented by Seidl (2006), and one could argue that it is the basis of the Partial CAD for QE problems by Collins and Hong (1991). Our contribution here is to explain how such cells can easily be identified in the presence of ECs. We demonstrate with our example.

Example 6. *Return to the problem from Examples 2 − 5 and in particular the CAD of* $\mathbb{R}^2$ *produced with 15 cells in Example 5. On 5 of these 15 cells the polynomial* R *is zero and on the others it is either positive or negative.*

Now, ϕ *can only be satisfied above the 5 cells, as elsewhere the two explicit EC defining polynomials cannot share a root and thus cannot vanish together. We can already conclude the truth value for the 10 cells (false) and thus we do not need to lift over them, except in the trivial sense of extending them to a cylinder in* $\mathbb{R}^3$*. Lifting over the 5 cells where* $R = 0$ *with respect to* f_1 *gives 15 cells, which when combined with the 10 cylinders gives a CAD of* $\mathbb{R}^3$ *with 25 cells that is truth-invariant for* ϕ*.*

Remark 6. *The improvements in this subsection are affecting the concluded structure of the output*[10]*. The final 25 cell truth-invariant CAD in Example 6 is* not *sign-invariant for* f_1*. The cylinders above the 10 cells in* $\mathbb{R}^2$ *where* $R \neq 0$ *may have* f_1 *varying sign, but since* f_1 *can never equal zero at the same time as* f_2 *in these cells it does not affect the truth of* ϕ*.*

Identifying the 5 cells in $\mathbb{R}^2$ where $R = 0$ was trivial since they are simply the sections of the second lift: those cells with second entry even in the cell index. Similarly, all sections in the third lift are those where f_1 is zero, however, we cannot conclude that f_2 is also zero on these as Theorem 2

[9] Although we note that if the recent work of McCallum et al. (2019) can be extended to ECs then such worries may be unnecessary altogether.

[10] In comparison with Remark 4.

only guarantees that f_2 is sign-invariant on them. So we must still finish by evaluating the polynomials at the sample points, but only for the sections.

Reducing the number of cells for stack generation clearly decreases output size, and since the cells can be identified using only an integer parity check computation time decreases (less real root isolation is performed). As described in Remark 5 for the improvements in Section 3.1, this also decreases the risk of non well-oriented input.

4. Algorithm

We present Algorithm 1 (note that it is split into two parts) to build a truth-invariant CAD for a formula through the use of ECs. The input is a quantifier free formula in $x_1, \ldots, x_n$ (we assume a fixed variable ordering). The output is a CAD of $\mathbb{R}^n$ which we interpret as a set of cells where each cell comes with a cell index and sample point. Each of these is a list of n numbers (recall Section 2.1 for how sample points and indices are represented and extended). This is the minimum information needed, but implementations may choose to store more, such as the formulae for cells.

The first two steps process the input formula into sets of polynomials: A_n contains all polynomials in the input formula; while the E_k are subsets of A_n which each contain the defining polynomial for a primitive EC of main variable x_k if one is available, and are empty otherwise. Step 2 is a simple extraction but Step 1 is non-trivial: it must identify suitable ECs for use in projection, and these may not be explicit in the formula (see Definition 2 and Example 1). In practice this will likely require EC propagation (as described in Section 2.6); and EC designation (choosing which of a variety of potential ECs with the same mvar to identify for reduced projection). We discuss the intricacies of this latter step in more detail later.

Steps $3 - 13$ run the projection phase of the algorithm. Each projection starts by identifying contents and primitive parts. This is not required for E_i: since we assume primitive ECs we have $\text{cont}(E_i) = \emptyset$, $\text{prim}(E_i) = E_i$ and the set of factors of E_i contained in B_i for each i.

When there is no declared EC (E_i is empty) the projection operator (5) is used (step 8). Otherwise the operator (7) is used (step 13), unless it is the very first or very last projection (step 11) when we use (6). This follows the theory detailed in Section 2. In each case the output of the projection is combined with the contents to form the next layer of projection polynomials.

Algorithm 1: CAD using multiple ECs (part 1 of 2)

Input : A QFF ϕ in variables $x_1, \ldots, x_n$

Output: Either: $\mathcal{D}$, a truth-invariant CAD of $\mathbb{R}^n$ for ϕ formed from a set of cells each defined by an index and a sample point; or **FAIL**, if not well-oriented.

1 Identify from ϕ a sequence of sets E_k, $k = 1, \ldots, n$, each either empty or containing a single primitive polynomial with mvar x_k, where each polynomial defines an EC for ϕ;
2 Extract the set of defining polynomials A_n;
3 **for** $k = n, \ldots, 2$ **do**
4 Set B_k to the finest squarefree basis for $\text{prim}(A_k)$;
5 Set C to $\text{cont}(A_k)$;
6 Set F_k to the finest squarefree basis for E_k;
7 **if** F_k *is empty* **then**
8 Set $A_{k-1} := C \cup P(B_k)$;
9 **else**
10 **if** $k = n$ *or* $k = 2$ **then**
11 Set $A_{k-1} := C \cup P_{F_i}(B_i)$;
12 **else**
13 Set $A_{k-1} := C \cup P^*_{F_i}(B_i)$;
14 If E_1 is not empty then set p to be its element; otherwise set p to the product of polynomials in A_1;
15 Build a CAD of the real line, $\mathcal{D}_1$, according to the real roots of p;
16 **if** $n = 1$ **then**
17 **return** $\mathcal{D}_1$;

Steps $14 - 17$ construct a CAD for the real line (returning it for univariate input), in what is called the *base phase*. If there is a declared EC in the smallest variable then the real line is decomposed according to its roots; otherwise according to the roots of all the univariate projection polynomials.

Steps $18 - 34$ run the lifting phase, incrementally building CADs of $\mathbb{R}^k$ for $k = 2, \ldots, n$. For each k there are two considerations:

Algorithm 1: CAD using multiple ECs (part 2 of 2)

```
18 for k = 2, ..., n do
19     Initialise D_k to be an empty set;
20     if F_k is empty then
21         Set L := B_k;
22     else
23         Set L := F_k;
24     if E_{k-1} is empty then
25         Set C_a := D_{k-1} and C_b empty;
26     else
27         Set C_a to be cells in D_{k-1} whose cell index final entry is even;
28         Set C_b := D_{k-1} \ C_a;
29     for each cell c ∈ C_a do
30         if An element of L is nullified over c then
31             return FAIL;
32         Generate a stack of cells over c with respect to the polynomials
             in L. Form new sample points and cell indicies as extensions
             of those from c;
33     for each cell c ∈ C_b do
34         Extend to a single cell in R^k (cylinder over c) (the extension to
             the index is simply 1 and the extension to the sample point
             can be any number);
35 return D_n.
```

- First, whether there is a declared EC with mvar x_k. If so we lift only with respect to this (step 23) and if not we use all projection polynomials with mvar x_k (step 21). See Section 3.1.
- Second, whether there is a declared EC with mvar x_{k-1}. If so we restrict stack generation to those cells where the EC was satisfied. These are simply those with the final entry of the cell index I_{k-1} even (step 27). We lift the other cells trivially to a cylinder in step 34. See Section 3.2.

Algorithm 1 clearly terminates. We will verify that it produces a truth-invariant CAD for the formula if the input is well-oriented, as defined below.

Definition 4. *For $k = 2, \ldots, n$ define sets:*

- L_k *– the* lifting polynomials*: defining polynomial of the declared EC with mvar x_k if it exists; else all projection polynomials with mvar x_k.*
- $\mathcal{C}_k$ *– the* lifting cells*: those cells in the CAD of $\mathbb{R}^{k-1}$ in which the designated EC with mvar x_{k-1} vanishes if it exists, and all cells in that CAD otherwise.*

The input of Algorithm 1 is well-oriented *if for $k = 2, \ldots, n$ no element of L_k is nullified over an element of $\mathcal{C}_k$.*

Theorem 4. *Algorithm 1 satisfies its specification.*

Proof. We must show the CAD is truth-invariant for ϕ, unless the input is not well-oriented when FAIL is returned.

First consider input with $n = 1$. The projection phase would not run, with the algorithm jumping to the CAD construction in step 14, returning the output in step 17. If there was no declared EC then the CAD is sign-invariant for all polynomials defining ϕ and thus every cell is truth invariant for ϕ. If there was a declared EC then the output is sign-invariant for its defining polynomial. Cells would either be intervals where the formula must be false; or points, where the EC is satisfied, and the formula either identically true or false depending on the signs of the other polynomials.

Next suppose that the input were not well-oriented (Definition 4). For a fixed k, the conditional in steps $20 - 23$ sets the lifting polynomials L_k to L and the conditional in steps $24 - 28$ the lifting cells $\mathcal{C}_k$ to $\mathcal{C}_a$. Thus it is exactly the conditions of Definition 4 which are checked by step 30, returning FAIL in step 31 when they are not satisfied. Hence if the lifting phase completes then the input is well-oriented.

From now on we suppose $n > 1$ and the input is well-oriented. For a fixed k with $2 \leq k \leq n$ define *admissible* cells to be those in the CAD $\mathcal{D}_{k-1}$ of $\mathbb{R}^{k-1}$ produced by Algorithm 1 where all declared ECs with mvar smaller than x_k are satisfied, or to be all cells in that CAD if there are no such ECs. Then let $I(k)$ be the following statement in italics.

Over admissible cells (in $\mathbb{R}^{k-1}$) the CAD $\mathcal{D}_k$ of $\mathbb{R}^k$ produced by Algorithm 1 is: (a) order-invariant for any EC with mvar x_k; (b) order- (sign- if $k = n$) invariant for all projection polynomials with mvar x_k on sections of the EC over admissible cells, or over all admissible cells if no EC exists.

We shall prove that, for all k with $1 \leq k \leq n$, $I(k)$ is true. We have already proved $I(1)$ (the induction base). Now let $1 < k \leq n$ and assume $I(k-1)$ as the induction hypothesis. The truth of $I(k)$, which completes the induction, is then a consequence of the following remarks:

- When E_k is empty we use Theorem 1 to assert all projection polynomials with mvar x_k are order-invariant in the stacks over admissible cells giving (a) and (b).
- When E_k is not empty and $k = 2$ we used the projection operator (6). Theorem 2 allows us to conclude (b) and that the EC is sign-invariant in admissible cells. The stronger property of order-invariance follows since the lifting polynomials form a squarefree basis in two variables.
- When E_k is not empty and $k = n$ we used the projection operator (6). Theorem 2 allows us to conclude (b), but also (a) since in the case $k = n$ the statement requires only sign-invariance.
- When E_k is not empty and $2 < k < n$ we used the projection operator (7). Theorem 3 explains that the additions (7) makes to (6) are sufficient to conclude the statement.

In each case the assumptions of the theorems are met by the inductive hypothesis exactly over admissible cells, according to whether E_{k-1} was empty.

From the definition of admissible cells, we know that ϕ is false (and thus trivially truth invariant) upon all cells in the CAD of $\mathbb{R}^n$ built over an inadmissible cell of $\mathbb{R}^k$, $k < n$. Coupled with the truth of (a) for $k = 1, \ldots, n$, this implies the CAD of $\mathbb{R}^n$ is truth-invariant for the conjunction of ECs (although it may not be truth-invariant for any one individually). The truth of (b) implies that on those cells where all ECs are satisfied, the other polynomials in ϕ are sign-invariant and thus ϕ is truth-invariant. □

5. Worked Example

We consider an example with sufficient variables to show all the features of the algorithm but still small enough to discuss in text. Assume variable ordering $z \succ y \succ x \succ u \succ v$ and define

$$f_1 := x - y + z^2, \qquad f_2 := z^2 - u^2 + v^2 - 1, \qquad g := x^2 - 1,$$
$$f_3 := x + y + z^2, \qquad f_4 := z^2 + u^2 - v^2 - 1, \qquad h := z.$$

We consider the formula

$$\phi = f_1 = 0 \wedge f_2 = 0 \wedge f_3 = 0 \wedge f_4 = 0 \wedge g \geq 0 \wedge h \geq 0.$$

The solution can be found manually by decomposing the system into blocks. The surfaces f_1 and f_3 are graphed in (x, y, z)-space on the left of Figure 2. They meet only on the plane $y = 0$ and this projection is shown on the right. The surfaces f_2 and f_4 are graphed in (z, u, v)-space on the left of Figure 3 and meet only when $z = \pm 1$. We consider only $z = +1$ due to $h \geq 0$, with this projection plotted on the right. We thus see that the solution set is

$$\{u = \pm v, x = -1, y = 0, z = 1\}.$$

To ascertain this by Algorithm 1 we must first propagate and designate ECs in Step 1. We choose to use f_1 first, calculate

$$\mathrm{res}_z(f_1, f_2) = (v^2 - u^2 + y - x - 1)^2$$

and assign $r_1 := v^2 - u^2 + y - x - 1$. So r_1 is the defining polynomial for an EC with mvar y. Similarly consider

$$\begin{aligned} \mathrm{res}_y\left(r_1, \mathrm{res}_z(f_1, f_3)\right) &= 16(u^2 - v^2 + x + 1)^4, \\ \mathrm{res}_y\left(r_1, \mathrm{res}_z(f_1, f_4)\right) &= 4(u^2 - v^2)^2 \end{aligned}$$

and assign $r_2 := u^2 - v^2 + x + 1$, $r_3 := u^2 - v^2$. These are defining polynomials for ECs with mvar x and u respectively. There is no series of resultants that leads to an EC with mvar v (they all result in constants by that stage). We hence identify $\{E_j\}_{k=1}^n := \{f_1\}, \{r_1\}, \{r_2\}, \{r_3\}, \{\,\}$ in Step 1.

The algorithm continues by extracting the defining polynomials

$$A_5 = \{f_1, f_2, f_3, f_4, g, h\}$$

and finds $B_5 = A_5, F_5 = E_5$ (in fact $F_i = E_i$ for all $i = 1, \ldots, 5$).

We now start the projection phase. There is a declared EC for the first projection so we use operator (6) to derive

$$\begin{aligned} A_4 := P_{F_5}(B_5) = \{&(x^2 - 1)^2, (-u^2 + v^2 - x + y - 1)^2, \\ &(u^2 - v^2 - x + y - 1)^2, 4y^2, x - y\}. \end{aligned}$$

Hence $C := \{x^2 - 1\}$ and

$$B_4 := \{y, y - x, -u^2 + v^2 - x + y - 1, u^2 - v^2 - x + y - 1\}.$$

For the next projection we must use operator (7), giving

$$A_3 := C \cup P^*_{F_4}(B_4) = \{x^2 - 1, u^2 - v^2 + x + 1, u^2 - v^2, u^2 - v^2 + 1\}.$$

Figure 2: The polynomials f_1 and f_3 from Section 5.

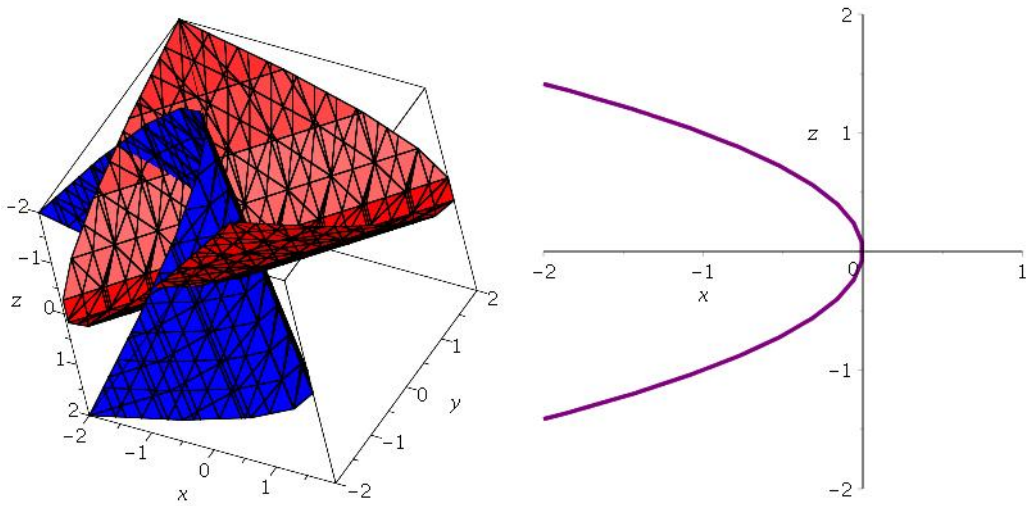
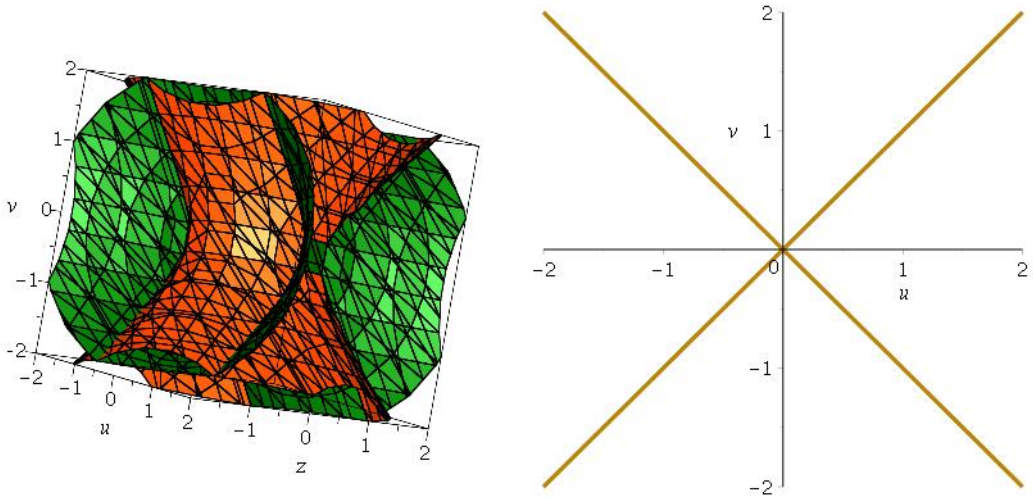


Figure 3: The polynomials f_2 and f_4 from Section 5.



For this example the extra discriminants in (7) all evaluated to constants and so could be discarded, while for all polynomials the leading coefficient was constant and so could be discarded with no further coefficients considered (see Remark 2). Then

$$B_3 := \{x^2 - 1, u^2 - v^2 + x + 1\}, \quad C := \{u^2 - v^2, u^2 - v^2 + 1\},$$

and the next projection also uses (7) to produce

$$A_2 := \{u^2 - v^2, u^2 - v^2 + 1, u^4 - 2u^2v^2 + v^4 + 2u^2 - 2v^2\}.$$

For the final projection there is no EC and so we use operator (5) to find $A_1 := \{v^2\}$. The base phase of the algorithm hence produces a 3-cell CAD of the real line isolating 0.

For the first lift we have $L = \{u^2 - v^2\}$ and C_a containing all 3 cells. Above the two intervals we split into 5 cells by the curves $u = \pm v$, while above $v = 0$ we split into three cells about the origin. From these 13 cells of $\mathbb{R}^2$ we select the 5 which were sections of $u^2 - v^2$ for C_a. These are lifted with respect to $L = \{r_2\}$, and the other 8 are simply extended to cylinders in $\mathbb{R}^3$. Together this gives a CAD of $\mathbb{R}^3$ with 23 cells. The next two lifts are similar, producing first a CAD of $\mathbb{R}^4$ with 53 cells and finally a CAD of $\mathbb{R}^5$ with 113 cells. The entire calculation takes less than a second in Maple.

5.1. Choice in EC designation

Algorithm 1 could have been initialised with alternative EC designations. There were the 4 explicit ECs with mvar z, and by taking repeated resultants we discover the following implicit ECs, in sets with decreasing mvar:

$$\begin{aligned}
&\{y^2, u^2 - v^2 + x - y + 1, -u^2 + v^2 + x - y + 1,\\
&\quad u^2 - v^2 + x + y + 1, -u^2 + v^2 + x + y + 1\},\\
&\{x + 1, -u^2 + v^2 + x + 1, u^2 - v^2 + x + 1\},\\
&\{u^2 - v^2\}.
\end{aligned}$$

There are hence 60 possible permutations of EC designation, but they lead to only 3 different output sizes: 113, 103 and 93 cells. Heuristics for other questions of CAD problem formulation (Dolzmann et al., 2004; Bradford et al., 2013b; Huang et al., 2014; Wilson et al., 2014) could perhaps be adapted to assist here. None of these are the minimal truth invariant CAD for ϕ as all split the CAD of $\mathbb{R}^1$ at $v = 0$ (from the discriminant $u^2 - v^2$).

5.2. Comparison with previous EC theory

A sign-invariant CAD of $\mathbb{R}^5$ for the 6 input polynomials can be produced by QEPCAD with 1,118,205 cells. Neither the `RegularChains` Library in MAPLE (Chen et al., 2009) nor our MAPLE package (England et al., 2014b) could do this in under an hour.

Our implementation of the algorithm by McCallum (1999b), which uses operator (6) once but also performs the final lift with respect to the EC only, can produce a CAD with either 3023, 10,935 or 48,299 (twice) cells depending on which EC is designated. The QEPCAD implementation of that algorithm gives 11,961, 30,233, 158,475, or 158,451 cells. Comparing these sets of figures we see the dramatic improvements from just a single reduced lift.

Allowing QEPCAD to propagate the four ECs (so a similar projection phase as Algorithm 1 but then a normal CAD lifting phase) produces a CAD with 21,079 cells. By declaring only a subset of the four (which presumably changes the designations of implicit ECs) a CAD with 5,633 cells can be produced, still much more than using Algorithm 1.

The `RegularChains` Library can also make use of multiple ECs[11], as detailed by Bradford et al. (2014), a CAD can be produced instantly. There are choices with analogies to designation (England et al., 2014a)), but they all lead to a 137 cell output. In particular, they all have an induced CAD of the real line which splits at $v = \pm 1$ as well as $v = 0$.

We note that our MAPLE implementation is unrefined and unoptimised. We do not claim it as a leading CAD implementation. The purpose of the paper is to illustrate the state of the art in CAD with EC theory, that all CAD implementations should adapt to reproduce. The worked example shows the clear benefits of the improved lifting techniques, which we next generalise with a complexity analysis.

6. Complexity Analysis of CAD with EC

We build on recent work by Bradford et al. (2016) to measure the dominant term in bounds on the number of CAD cells produced. Numerous studies have shown this to be closely correlated to the computation time (Dolzmann et al., 2004; Bradford et al., 2013a, 2014). We assume CAD input with m polynomials of maximum degree d in any one of n variables.

[11] but only the latest version from `www.regularchains.org`

Definition 5. *Consider a set of polynomials p_j. The* combined degree *of the set is the maximum degree (taken with respect to each variable) of the product of all the polynomials in the set:* $\max_i(\deg_{x_i}(\prod_j p_j))$.

The set has the (m,d)-property *if it may be partitioned into m subsets, each with maximum combined degree d.*

For example, $\{y^2 - x, y^2 + 1\}$ has combined degree 4 and thus the $(1, 4)$-property, but also the $(2, 2)$-property.

We will measure complexity by keeping track of the number and degree of projection polynomials. Of course, by replacing $\{f, g\}$ with $\{fg\}$ we can reduce the number at the cost of increasing the degree, but since it is much easier to find the roots of $\{f, g\}$ than $\{fg\}$, we do not want to do that. The (m, d)-property, introduced in the thesis of McCallum (1985), is in some sense the optimal measure of these properties.

Bradford et al. (2016) proved that if A has the (m, d)-property then $P(A) \cup \text{cont}(A)$ has the $(M, 2d^2)$-property with $M = \left\lfloor \frac{1}{2}(m + 1)^2 \right\rfloor$. When $m > 1$, we can bound M by m^2 (but we need $2m^2$ to cover $m = 1$).

6.1. Complexity of sign-invariant CAD

If A has the (m, d)-property then so does its squarefree basis. Hence applying this result recursively (as in Table 1) measures the growth in (m, d)-property during projection under operator (5). After the first projection there are multiple polynomials and so the tighter bound for M is used.

Table 1: Projection under operator (5).

Variables	Number	Degree
n	m	d
$n-1$	$2m^2$	$2d^2$
$n-2$	$4m^4$	$8d^4$
$\vdots$	$\vdots$	$\vdots$
$n-r$	$2^{2^{r-1}}m^{2^r}$	$2^{2^r-1}d^{2^r}$
$\vdots$	$\vdots$	$\vdots$
1	$2^{2^{n-2}}m^{2^{n-1}}$	$2^{2^{n-1}-1}d^{2^{n-1}}$

The number of real roots in a set with the (m, d)-property is at most md (although in practice many will be in $\mathbb{C} \setminus \mathbb{R}$). The number of cells in the

CAD of $\mathbb{R}^1$ is thus bounded by twice the product of the final two entries, plus 1. Similarly, if we let d_i and m_i be the entries in the Number and Degree columns of Table 1 from the row corresponding to i variables, then the total number of cells in the CAD of $\mathbb{R}^n$ is bounded by

$$\prod_{i=1}^{n} [2m_i d_i + 1] = (2md + 1) \prod_{r=1}^{n-1} \left[2 \left(2^{2^{r-1}} m^{2^r} \right) \left(2^{2^r - 1} d^{2^r} \right) + 1 \right]. \tag{8}$$

Omitting the +1s will leave us with the dominant term of the bound, which evaluates to give the following result.

Theorem 5. *The dominant term in the bound on the number of CAD cells in $\mathbb{R}^n$ produced using (5) is*

$$(2d)^{2^n - 1} m^{2^n - 1} 2^{2^{n-1} - 1}. \tag{9}$$

6.2. Reduced projection from ECs

From now on assume ℓ equational constraints, $0 < \ell \leq \min(m, n)$, all with different mvar. For simplicity we assume these variables are $x_n, \ldots, x_{n-\ell+1}$, i.e. the first ℓ projections are the reduced ones.

Lemma 6. *Suppose A is a set with the (m, d)-property and $E \subset A$ has the $(1, d)$-property. Then $\mathrm{cont}(A) \cup P_E^*(A)$ has the $(3m, 2d^2)$-property.*

Proof. Bradford et al. (2016) proved that applying $P_E(A) \cup \mathrm{cont}(A)$ gives a set of polynomials of size at most $\left\lfloor \frac{1}{2}(3m + 1) \right\rfloor$ with combined degree $2d^2$.

We now have the additional discriminant and coefficients of (7) to take care of. Each polynomial in $A \setminus E$ will generate an additional discriminant of degree at most $d(d - 1)$ and $(d + 1)$ additional coefficients of degree at most d. Multiplying all these polynomials together gives a single polynomial of degree at most $2d^2$. There are $m - 1$ polynomials in $A \setminus E$ and so in total this projection generates

$$\left\lfloor \tfrac{1}{2}(3m + 1) \right\rfloor + (m - 1) < 3m$$

polynomials of degree $2d^2$. □

We apply this recursively in the top part of Table 2, with the bottom derived via the process for P, as in Table 1.

Table 2: Projection with (7) ℓ times and then (5).

Variables	Number	Degree
n	m	d
$n-1$	$3m$	$2d^2$
$\vdots$	$\vdots$	$\vdots$
$n-\ell$	$3^\ell m$	$2^{2^\ell-1}d^{2^\ell}$
$n-(\ell+1)$	$3^{2\ell}m^2$	$2^{2^{\ell+1}-1}d^{2^{\ell+1}}$
$\vdots$	$\vdots$	$\vdots$
$n-(\ell+r)$	$3^{2^r\ell}m^{2^r}$	$2^{2^{\ell+r}-1}d^{2^{\ell+r}}$
$\vdots$	$\vdots$	$\vdots$
1	$3^{2^{(n-1-\ell)}\ell}m^{2^{n-1-\ell}}$	$2^{2^{n-1}-1}d^{2^{n-1}}$

Define d_i and m_i as the entries in the Number and Degree columns of Table 2 from the row corresponding to i variables. We can bound the number of real roots of projection polynomials in i variables by $m_i d_i$. If we lifted with respect to all projection polynomials the cell count would be bounded by

$$\begin{aligned}\prod_{i=1}^{n}\left[2m_id_i+1\right] &= \prod_{i=1}^{n-(\ell+1)}\left[2m_id_i+1\right]\times\prod_{i=n-\ell}^{n}\left[2m_id_i+1\right] \qquad (10)\\ &= \prod_{s=0}^{\ell}\left[2\left(3^sm\right)\left(2^{2^s-1}d^{2^s}\right)+1\right]\times\prod_{r=1}^{n-\ell-1}\left[2\left(3^{2^r\ell}m^{2^r}\right)\left(2^{2^{\ell+r}-1}d^{2^{\ell+r}}\right)+1\right].\end{aligned}$$

Omitting the +1 from each product allows us to calculate the dominant term of the bound explicitly as

$$(2d)^{2^n-1}m^{2^{n-\ell}+\ell-1}3^{\ell 2^{n-\ell}+\ell(\ell-3)/2}. \qquad (11)$$

6.3. Reduced lifting from ECs

Now we consider the benefit of improved lifting. Start by considering the CAD of $\mathbb{R}^{n-(\ell+1)}$. There can be no reduced lifting until this point and so the cell count bound is given by the second product in (10), which we will denote by (†). The lift to $\mathbb{R}^{n-\ell}$ will involve stack generation over all cells, but only with respect to the EC. This can have at most $d_{n-\ell}$ real roots and so the CAD at most $(2d_{n-\ell}+1)\times(\dagger)$ cells.

The next lift, to $\mathbb{R}^{n-\ell-1}$, will lift the sections with respect to the EC, and the sectors only trivially (to produce the same number of cylinders). Hence the cell count bound is

$$(2d_{n-(\ell-1)}+1)(d_{n-\ell})(\dagger)+(d_{n-\ell}+1)(\dagger)$$

with dominant term $2d_{n-(\ell-1)}d_{n-\ell}(\dagger)$. Subsequent lifts follow the same pattern and so $2d_nd_{n-1}\dots d_{n-(\ell-1)}d_{n-\ell}(\dagger)$ is the dominant term in the bound for $\mathbb{R}^n$. This evaluates to give the following result.

Theorem 7. *Consider the CAD of $\mathbb{R}^n$ produced using Algorithm 1 in the presence of ECs in the top ℓ variables of the ordering. The dominant term in the bound on the number of cells is*

$$\begin{aligned}&2\prod_{s=0}^{\ell}\left[2^{2^s-1}d^{2^s}\right]\prod_{r=1}^{n-\ell-1}\left[2\left(2^{2^r\ell}m^{2^r}2^{2^{\ell+r}-1}d^{2^{\ell+r}}\right)\right]\\&=(2d)^{2^n-1}m^{2^{n-\ell}-2}2^{-\ell}3^{\ell 2^{n-\ell}-2\ell}.\end{aligned}\tag{12}$$

6.4. Summary of complexity analysis

The bound in Theorem 7 is strictly less than the one in Theorem 5. The double exponent of m has decreased by the number of ECs; the result of the improved projection in (11). Then improved lifting has reduced the single exponents in the bound further still in (12).

However, even with this maximal use of ECs, CAD is still doubly exponential in the number of variables due to the first term in (12), the one whose degree is the degree term. This should not be surprising: the theory of ECs is based around reducing the number of polynomials identified in each projection, but not the number of projections which controls the degree growth. Indeed, we can see directly from Tables 1 and 2 that at the end of projection we are dealing with univariate polynomials of degree doubly exponential in n regardless of whether we used ECs or not. Reduced lifting allows us to avoid isolating the real roots of many of these polynomials, but we will always need to consider at least one (the EC defining polynomial). To control degree growth we must show this to be of a lower degree.

7. Controlling Degree Growth

7.1. Degree growth through iterated resultant calculations

The doubly exponential degree comes from the use of iterated resultant calculations during projection: the resultant of two degree d polynomials is

the determinant of a $2d \times 2d$ matrix whose entries all have degree at most d, and thus a polynomial of degree at most $2d^2$. This increase in degree compounded by $(n-1)$ projections gives the first term of the bound (9). Note that the derivation of ECs themselves via EC propagation (Section 2.6) is itself such an iterated resultant calculation. So even though the EC theory of the previous sections allows us to avoid constructing or lifting with many such polynomials, the ECs themselves encode the degree.

The purpose of the resultant in CAD construction is to ensure that the points in lower dimensional space where polynomials vanish together are identified, and thus that the behaviour over a sample point in a lower dimensional cell is indicative of the behaviour over the cell as a whole. The iterated resultant (and discriminant) calculations involved in CAD have been studied previously, for example by McCallum (1999a) and Lazard and McCallum (2009). We will follow the work of Busé and Mourrain (2009) who consider the iterative application of the univariate resultant to multivariate polynomials, demonstrating decompositions into irreducible factors involving the multivariate resultants[12]. They show that the approach will identify polynomials of higher degree than the true multivariate resultant and thus more than required for the purpose of identifying implicit equational constraints. For example, given 3 polynomials in 3 variables of degree d the true multivariate resultant has degree $\mathcal{O}(d^3)$ rather than $\mathcal{O}(d^4)$.

The key result of Busé and Mourrain (2009) for our purposes follows. Note that this considers polynomials of a given *total degree*. However, the CAD complexity analysis discussed above and later is (following previous work on the topic) with regards to polynomials of *degree at most d* in a given variable. For clarity we use the Fraktur font when discussing total degree and Roman fonts when the maximum degree.

Corollary 8 (Busé and Mourrain (2009, Cor. 3.4))**.** *Given three polynomials* $f_k(\mathbf{x}, y, z)$ *of the form*

$$f_k(\mathbf{x}, y, z) = \sum_{|\alpha|+i+j \leq \mathfrak{d}_k} a^{(k)}_{\alpha,i,j} \mathbf{x}^\alpha y^i z^j \in S[\mathbf{x}][y, z],$$

where S *is any commutative ring, then the iterated univariate resultant*

$$\operatorname{res}_y \left(\operatorname{res}_z(f_1, f_2), \operatorname{res}_z(f_1, f_3) \right) \in S[\mathbf{x}]$$

[12]They follow the formalisation of Jouanolou (1991) as laid out in (Busé and Mourrain, 2009, §2).

is of total degree at most $\mathfrak{d}_1^2\mathfrak{d}_2\mathfrak{d}_3$ *in* $\mathbf{x}$*, and we may express it in multivariate resultants (Jouanolou, 1991), denoted* Res*, as*

$$\begin{aligned}\operatorname{res}_y\big(\operatorname{res}_z(f_1,f_2),\operatorname{res}_z(f_1,f_3)\big) &= (-1)^{\mathfrak{d}_1\mathfrak{d}_2\mathfrak{d}_3}\operatorname{Res}_{y,z}(f_1,f_2,f_3)\\ &\times \operatorname{Res}_{y,z,z'}\big(f_1(\mathbf{x},y,z),f_2(\mathbf{x},y,z),f_3(\mathbf{x},y,z'),\delta_{z,z'}(f_1)\big).\end{aligned} \tag{13}$$

Moreover, if the polynomials f_1, f_2, f_3 *are sufficiently generic and* $n > 1$*, then this iterated resultant has exactly total degree* $\mathfrak{d}_1^2\mathfrak{d}_2\mathfrak{d}_3$ *in* $\mathbf{x}$ *and both resultants on the right hand side of the above equality are distinct and irreducible.*

Remark 7. *Although not stated as part of the result by Busé and Mourrain (2009), under these generality assumptions,* $\operatorname{Res}_{y,z}(f_1,f_2,f_3)$ *has total degree* $\mathfrak{d}_1\mathfrak{d}_2\mathfrak{d}_3$ *and the second resultant on the right hand side of (13) has total degree* $\mathfrak{d}_1(\mathfrak{d}_1-1)\mathfrak{d}_2\mathfrak{d}_3$ *(see (Busé and Mourrain, 2009, Proposition 3.3) and (McCallum, 1999a, Theorem 2.6)).*

Busé and Mourrain (2009) interpret this result in the following quote:[13].

> The resultant $r_{12} := \operatorname{res}_z(f_1,f_2)$ defines the projection of the intersection curve between the two surfaces $\{f_1=0\}$ and $\{f_2=0\}$. Similarly, $r_{13} := \operatorname{res}_z(f_1,f_3)$ defines the projection of the intersection curve between the two surfaces $\{f_1=0\}$ and $\{f_3=0\}$. Then the roots of $\operatorname{res}_y(r_{12},r_{13})$ can be decomposed into two distinct sets: the set of roots x_0 such that there exists y_0 and z_0 such that
>
> $$f_1(x_0,y_0,z_0) = f_2(x_0,y_0,z_0) = f_3(x_0,y_0,z_0),$$
>
> and the set of roots x_1 such that there exist two distinct points (x_1,y_1,z_1) and (x_1,y_1,z_1') such that
>
> $$f_1(x_1,y_1,z_1) = f_2(x_1,y_1,z_1) \quad \text{and} \quad f_1(x_1,y_1,z_1') = f_3(x_1,y_1,z_1').$$
>
> The first set gives rise to the term $\operatorname{Res}_{y,z}(f_1,f_2,f_3)$ in the factorization of the iterated resultant $\operatorname{res}_y(\operatorname{res}_{12},\operatorname{res}_{13})$, and the second set of roots corresponds to the second factor.

If the f_i are all ECs then only the first set are of interest to us as the truth of the formula of interest needs them all to vanish at once. However, for a general CAD construction, the second set of roots may also be necessary as they indicate points where the geometry of the sectors changes.

[13] The quote contains a correction in the description of the second set of roots (removing a dash from y_1 in the second distinct point). The mistake was identified by the anonymous referees of (England and Davenport, 2016).

7.2. How large are these resultants?

Consider three ECs defined by f_1, f_2 and f_3 of degree at most d in each variable *separately*; and that we wish to eliminate two variables $z = x_n$ and $y = x_{n-1}$. We may naïvely set each $\mathfrak{d}_i = nd$ to bound the total degree.

The following approach does better. Let $K = S[x_1, \ldots, x_{n-2}, y, z]$ and $L = S[\xi_1, \ldots, \xi_N, y, z]$. Only a finite number of monomials in $x_1, \ldots, x_{n-2}$ occur as coefficients of the powers of y, z in f_1, f_2 and f_3. Map each such monomial $x^\alpha = \prod_{i=1}^{n-2} x_i^{\alpha_i}$ to $\widetilde{m_j} := \xi_j^{\max \alpha_i}$ (using a different ξ_j for each monomial[14]) and let $\widetilde{f_i} \in L$ be the result of applying this map to the monomials in f_i. Operation $\widetilde{\ }$ commutes with taking resultants in y and z (but not x_i).

The total degree in the ξ_j of $\widetilde{f_i}$ is the same as the maximum degree in all the $x_1, \ldots, x_{n-2}$ of f_i, i.e. bounded by d, and hence the total degree of the $\widetilde{f_i}$ in all variables is bounded by $3d$ (d for the ξ_i, d for y and d for z). If we apply (13) to the $\widetilde{f_i}$, we see that $\operatorname{res}_y\big(\operatorname{res}_z(\widetilde{f_1}, \widetilde{f_2}), \operatorname{res}_z(\widetilde{f_1}, \widetilde{f_3})\big)$ has a factor $\operatorname{Res}_{y,z}(\widetilde{f_1}, \widetilde{f_2}, \widetilde{f_3})$ of total degree (in the ξ_j) $(3d)^3$. Hence, by inverting $\widetilde{\ }$, we may conclude $\operatorname{Res}_{y,z}(f_1, f_2, f_3)$ has maximum degree, in each x_i, of $(3d)^3$.

The results of Jouanolou (1991) and Busé and Mourrain (2009) apply to any number of eliminations. In particular, if we have eliminated not 2 but $\ell - 1$ variables we will have a polynomial $\operatorname{Res}_{x_{n-\ell+1}\ldots x_n}(f_{n-\ell}, \ldots, f_n)$ of maximum degree $\ell^\ell d^\ell$ in the remaining variables $x_1, \ldots, x_{n-\ell}$ as the last implicit EC. Therefore the multivariate resultants we need, $\operatorname{Res}_{x_{n-\ell+1}\ldots x_n}$, only have singly-exponential growth, rather than the doubly-exponential growth of the iterated resultants: can we compute them?

7.3. Gröbner basis instead of iterated resultants

A *Gröbner Basis* G is a particular generating set of an ideal I (within the ring of polynomials over an algebraically closed field) 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 G. Gröbner Bases (GB) allow 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. Their properties and an algorithm to derive a GB for any ideal were introduced in the 1965 PhD thesis of Buchberger (2006) (since republished). There has been much research to improve and optimise GB

[14] We could economise: if $x_1 x_2^2 \mapsto \xi_1^2$, then we could map $x_1^2 x_2^4$ to ξ_1^4 rather than a new ξ_2^4. Since this is for the analysis and not in implementation, we ignore such possibilities.

calculation, with the F_5 algorithm of Faugère (2002) perhaps the most used approach currently.

Like CAD the calculation of a GB is necessarily doubly exponential in the worst case (Mayr and Meyer, 1982) (with lexicographic monomial ordering). Recent work by Mayr and Ritscher (2013) showed that rather than being doubly exponential with respect to the number of variables present the dependency is in fact on the dimension of the ideal. Despite this bound GB computation can often be done very quickly usually to the point of instantaneous for any problem tractable by CAD, as demonstrated for example by Wilson et al. (2012).

A reasonably common CAD technique is to precondition systems with multiple ECs by replacing the ECs by their GB. I.e. let $E = \{e_1, e_2, \dots\}$ be a set of polynomials; $G = \{g_1, g_2, \dots\}$ a GB for E; and B any Boolean combination of constraints, $f_i \, \sigma_i \, 0$, where $\sigma_i \in \{<, >, \leq, \geq, \neq, =\}$) and $F = \{f_1, f_2, \dots\}$ is another set of polynomials. Then

$$\Phi := (e_1 = 0 \wedge e_2 = 0 \wedge \dots) \wedge B \quad \text{and} \quad \Psi := (g_1 = 0 \wedge g_2 = 0 \wedge \dots) \wedge B$$

are equivalent. A truth-invariant CAD for Ψ is also truth-invariant for Φ.

If we consider GB preconditioning of CAD in the knowledge of the improved projection schemes for ECs then we see an additional benefit. It provides implicit ECs which are not in the main variable of the system removing the need for EC propagation. Since our aim is to produce one EC in each of the last ℓ variables, we need to choose an ordering on monomials which is lexicographic with respect to $x_n \succ x_{n-1} \succ \dots \succ x_{n-\ell+1}$: it does not actually matter (in regards to the theory) how we tie-break after that[15].

Let us suppose that we have ℓ ECs $f_1, \dots, f_\ell$ (at least one of them, say f_1 must include x_n, and similarly we can assume f_2 includes x_{n-1} and so on), such that these imply (even over $\mathbb{C}$) that the last ℓ variables are determined (not necessarily uniquely) by the values of $x_1, \dots, x_{n-\ell}$. Then the vanishing of polynomials f_1, $\text{Res}_{x_n}(f_1, f_2)$, $\text{Res}_{x_n, x_{n-1}}(f_1, f_2, f_3)$ etc. are all implied by the ECs. Hence either they are in the GB, or they are reduced to 0 by the GB, which implies that smaller polynomials are in the GB. Hence our GB will contain polynomials (which are ECs) of degree (in each variable separately) at most

$$d,\ 4d^2,\ 27d^3,\ \dots,\ ((\ell+1)d)^{\ell+1}.$$

[15]Research suggests that 'total degree reverse lexicographic in the rest' is most efficient.

Note that we are not making, and in the light of the work Mayr and Ritscher (2013) cannot make, any similar claim about the polynomials in fewer variables. Also, it is vital that the ECs be in the last variables for our use of the work of Jouanolou (1991) and Busé and Mourrain (2009) to work. So our results do not directly extend from the case we study, first applying ℓ reduced CAD projections in the presence of ECs before reverting to standard projection), to the more general case of having any ℓ of the projections be reduced.

7.4. Inclusion in Algorithm 1

There are two routes to include the above suggestion in Algorithm 1.

1. Directly replace the explicit ECs in ϕ by those from the GB as suggested above. This is a pre-processing of the input to Algorithm 1. The identification of ECs in Step 1 involves only a minimal designation choice when there are multiple explicit ECs in ϕ with the same mvar.
2. Encode this process into a sub-algorithm for Step 1. The GB polynomials become additional options for designated ECs along with those from EC propagation and choices are made based on minimal degree or some other criteria (Wilson et al., 2012; Huang et al., 2016). However, in this case, if GB polynomials are designated they must be added to the input set A_n (a reinterpreting of Step 2 so it extracts both from ϕ and $\{E_k\}_{k=1}^{n}$).

The first approach is the one commonly used in implementations (and the one assumed in later discussions). The benefit of the second is that it caps any increase in the number of polynomials from the use of GBs.

It is unlikely that the GB would produce more polynomials in the main variable than explicit ECs (since we are starting with a generating set all in the main variable and deriving another which would mostly not be) but we have yet to rule it out. Of course, the number of polynomials in the input can bear little relation to the number generated by projection. But with the second approach any increase in the initial m is capped to the number of additional designated ECs taken from the GB. The second option may become preferable in the event of development of a good (cheap) heuristic.

8. Evaluating the Use of GBs for ECs

8.1. Worked Example

Let us work with variable ordering $z \succ y \succ x \succ w$; polynomials

$$f_1 := xy - z^2 - w^2 + 2z, \qquad f_2 := x^2 + y^2 + z^2 + w + z,$$
$$f_3 := -w^2 - y^2 - z^2 + x + z \qquad h := z + w;$$

and the following QFF for which we seek a truth-invariant CAD.

$$\phi := f_1 = 0 \wedge f_2 = 0 \wedge f_3 = 0 \wedge h > 0.$$

In theory, we could analyse this system with a sign-invariant CAD for the four polynomials however none of the CAD implementations in Maple could do this within 30 minutes. Instead, let us take advantage of the ECs. There are 3 explicit ECs all with mvar z meaning only one can be designated for the first projection. We can propagate to find additional implicit ECs:

$$\begin{aligned}
r_1 &= \mathrm{res}_z(f_1, f_2) = y^4 + 2xy^3 + (3x^2 - 2w^2 + 2w + 6)y^2 + (2x^3 - 2w^2x \\
&\quad + 2wx - 3x)y + x^4 - 2w^2x^2 + 2wx^2 + 6x^2 + w^4 - 2w^3 + 4w^2 + 6w, \\
r_2 &= \mathrm{res}_z(f_1, f_3) = y^4 + 2xy^3 + (x^2 - 2x + 2)y^2 + (x - 2x^2)y + w^2 + x^2 - 2x \\
r_3 &= \mathrm{res}_z(f_2, f_3) = 4y^2 + x^4 + 2x^3 - 2w^2x^2 + 2wx^2 + 3x^2 - 2w^2x + 2wx \\
&\quad - 2x + w^4 - 2w^3 + 3w^2 + 2w;
\end{aligned}$$

all with main variable y. Continuing the propagation with

$$R_1 := \mathrm{res}_y(r_1, r_2), \qquad R_2 := \mathrm{res}_y(r_1, r_3), \qquad R_3 := \mathrm{res}_y(r_2, r_3);$$

gives the three polynomials in the Appendix, each degree 16 in x. These are different polynomials[16] but a numerical plot shows them all to have overlapping real part. All possible resultants to eliminate x evaluate to 0.

Step 1 could hence produce $3 \times 3 \times 3 = 27$ possible configurations if ECs are identified by propagation. Our implementation could build CADs for only 6 of these configurations[17], when using a time limit of 30 minutes. Of the 6 completed there was an average of 423 cells calculated in 113 seconds.

[16] most easily verified by comparing the final lines of each.

[17] The common factor of these 6 was the designation of r_2 for second projection.

The optimal configuration gave 227 cells in 36 seconds using a designation of f_2, r_3 and R_2.

Now consider instead taking a GB of $\{f_1, f_2, f_3\}$. We use a plex monomial ordering on the same variable ordering as the CAD to achieve a basis:

$$
\begin{aligned}
g_1 &= 2z + x^2 + x - w^2 + w,\\
g_2 &= 4y^2 + x^4 + 2x^3 + (-2w^2 + 2w + 3)x^2 + (2w^2 + 2w - 2)x\\
&\quad + w^4 - 2w^3 + 3w^2 + 2w,\\
g_3 &= 4yx - x^4 - 2x^3 + (2w^2 - 2w - 5)x^2 + (2w^2 - 2w - 4)x\\
&\quad - w^4 + 2w^3 - w^2 - 4w,\\
g_4 &= (4w^4 - 8w^3 + 4w^2 + 16w)y + x^7 + 4x^6 + (-4w^2 + 4w + 18)x^5\\
&\quad + (-12w^2 + 12w + 36)x^4 + (5w^4 - 10w^3 - 31w^2 + 40w + 53)x^3\\
&\quad + (10w^4 - 20w^3 - 34w^2 + 52w + 32)x^2 - (2w^6 - 6w^5 - 7w^4 + 32w^3\\
&\quad - 13w^2 - 44w - 16)x - 2w^6 + 6w^5 - 2w^4 - 14w^3 + 12w^2 + 16w,\\
g_5 &= x^8 + 4x^7 + (-4w^2 + 4w + 18)x^6 + (-12w^2 + 12w + 36)x^5 + (6w^4\\
&\quad - 12w^3 - 30w^2 + 44w + 53)x^4 + 4(3w^4 - 6w^3 - 8w^2 + 15w + 8)x^3\\
&\quad + (-4w^6 + 12w^5 + 6w^4 - 48w^3 + 26w^2 + 64w + 16)x^2\\
&\quad + (-4w^6 + 12w^5 - 4w^4 - 28w^3 + 24w^2 + 32w)x\\
&\quad + w^8 - 4w^7 + 6w^6 + 4w^5 - 15w^4 + 8w^3 + 16w^2.
\end{aligned}
$$

This is an alternative generating set for the ideal defined by the explicit ECs and thus all $g_i = 0$ are ECs for ϕ. Note that the degrees of the GB polynomials (with respect to any one variable) are on average lower (and never greater) than those of the (corresponding) iterated resultants.

Deriving ECs this way removes the choice for EC with mvar z or x but there are 3 possibilities for the designation with mvar y. Designating g_2 yields 83 cells while either g_3 or g_4 result in 55 cells. All 3 configurations took less than 20 seconds to compute (with designating g_4 the quickest).

8.2. Effect on the complexity bound

We now consider how using a GB to produce the designated ECs will improve the complexity analysis of Section 6. The number of polynomials will be the same as found earlier in Table 2. But we must now track separately the degree of the designated EC and the degree of the main projection polynomials as they are derived differently. For simplicity we will ignore the constant term and focus on the exponents.

As described above, the designated ECs will have degrees $d, 4d^2, 27d^3, \ldots$ as tracked in the middle column of Table 3. For the projection polynomials in the top half of the table the reduced projection operator $P_F(B)$ will take discriminants and coefficients of the EC polynomial; and resultants of them with the other projection polynomials. Thus the highest degree polynomial produced will have degree that is the sum of the degree of the EC polynomial and the highest degree other polynomial. This generates the right column of Table 3. We see that the degree exponents here form the so called *Lazy Caterer's sequence*[18] otherwise known as the *Central Polygonal Numbers*. The remaining projections recorded in the bottom half of the table use the sign-invariant projection operator and so the degree is squared each time.

Table 3: Maximum degree of projection polynomials produced for CAD when using projection operator (7) for the first ℓ projections and then (5) for the remaining.

Variables	Maximum Degree	
	EC	Others
n	d	d
$n-1$	$4d^2$	d^2
$n-2$	$27d^3$	d^4
$n-3$	$256d^4$	d^7
$\vdots$	$\vdots$	$\vdots$
$n-\ell$	$\ell^\ell d^{\ell+1}$	$d^{\ell(\ell+1)(1/2)+1}$
$n-(\ell+1)$	$d^{\ell(\ell+1)+2}$	
$n-(\ell+2)$	$d^{2\ell(\ell+1)+2^2}$	
$n-(\ell+3)$	$d^{2^2\ell(\ell+1)+2^3}$	
$\vdots$	$\vdots$	
$n-(\ell+r)$	$d^{2^{r-1}\ell(\ell+1)+2^r}$	
$\vdots$	$\vdots$	
1	$d^{2^{n-\ell-2}\ell(\ell+1)+2^{n-\ell-1}}$	

Now let us use the the top line of equation (10) derived earlier as the

[18]The On-Line Encyclopedia of Integer Sequences (2010), Sequence Number A000124, https://oeis.org/A000124

bound when using improved EC projection and lifting before applying the degrees of the projection polynomials. We can substitute here with the degrees from Table 2 as the d_i. The term with base d may be computed by

$$\prod_{s=0}^{\ell} \left(\ell^{\ell} d^{s+1}\right) \prod_{r=1}^{n-\ell-1} \left(d^{2^{r-1}\ell(\ell+1)+2^r}\right).$$

The exponent of d evaluates to

$$2^{(n-\ell)}\tfrac{1}{2}(\ell^2+\ell+2) - \tfrac{1}{2}(\ell^2+\ell) - 2. \tag{14}$$

8.2.1. The ignored constants

Above, we tracked only the degree of the monomial in the bound, and not the constants that multiply it. As well as for simplicity, this was because we could not find a closed form expression for the product of constants generated. However, it is simple to check that the constant factors derived by the GB grow exponentially in ℓ while those from iterated resultants grow doubly exponentially. Further, the constant term can be shown to be strictly lower for all but the first few projections. Finally, note that in Section 7.1 we saw that the multivariate resultant was itself a factor of the iterated resultant.

8.2.2. Comparison with base m term

Let us compare the derived exponent (14) with that for the term with base m from (12): $2^{n-\ell} - 2$. We see that both show the double exponent of the complexity bound reducing by ℓ, the number of ECs used. However, the reduction in degree is not quite as clean as the exponential term in the single exponent is multiplied by a quadratic in ℓ. This is to be expected as the singly exponential dependency on ℓ in the Number column of Table 1 was only in the term with constant base while for Table 2 the term with base d is itself single exponential in ℓ.

8.3. Should one always use GBs?

In Section 8.1 we showed the significant savings available if one derived ECs with GBs and in Section 8.2 we showed this follows through into a theoretical lowering of the worse case complexity bound. The latter offers the first theoretical justification for what is a widely used CAD optimisation.

However, experimental studies by Buchberger and Hong (1991); Wilson et al. (2012); Huang et al. (2016) have shown that it is not *always* beneficial to pre-process CAD with GB. The most recent experiment by Huang et al. (2016) found that 75% of a data set of 1200 randomly generated CAD

problems benefited from GB preconditioning. So it is certainly worth giving consideration to how ECs are derived. As noted earlier, the cost of computing the GB itself is usually negligible in comparison to the CAD so it is reasonable to first compute the GB and then decide whether or not to use it. A simple man-made heuristic was presented by Wilson et al. (2012) to make the decision while Huang et al. (2016) described the training of a machine learning classifier to decide.

There is no contradiction here with the complexity analysis above: the analysis is for the worst case and large input and makes no claim to the average complexity or what happens for smaller input. However, we hypothesise that repeating those studies using the new multiple EC technology would see a reduction in the cases where GB hindered CAD.

9. Caveats and the Need for Primitivity

There are a few caveats to the results presented above. First, Algorithm 1 can fail for non-well oriented input, but as noted earlier, this restriction may be lifted if the new theory for Lazard's projection operator validated by McCallum et al. (2019) can be extended to the EC case. Second, the complexity analysis (both in Section 6 and 8.2) assumes the designated ECs are in strict succession at the start of projection. For the first analysis it was only made to simplify the working, but for the second analysis it was crucial. However, Algorithm 1 itself does not carry this restriction and savings will still clearly be made in this case.

The only substantial restriction in the paper is that the designated ECs[19] be defined by primitive polynomials in the main variable of the projection. The restriction is common in the literature and present in all the underlying theory of McCallum (1999b, 2001).

There are analogies to be made with the well-oriented issue (when a projection factor is nullified). Non-primitive projection factors are not a problem for general CAD because we can factorize prior to projection (order-invariance of factors implies order-invariance of the product). We cannot do the same for a non-primitive EC though, as the next example shows. Also, unlike the well-orientedness issue, the primitivity restriction it is not likely to be removed by developing a Lazard family of EC projection operators.

[19] whether they be explicit in the formula or calculated via iterated results or GBs.

9.1. Possibilities to use non-primitive ECs?

Example 7. *Consider $\phi := zy = 0 \wedge \varphi$. under ordering $\cdots \succ z \succ y \succ \ldots$. Polynomial zy is not primitive, so Algorithm 1 cannot use the explicit EC.*

We may be tempted to take $E = \{z\}$ as the primitive part, project with operator (6) and include the content y in the first projection. The CAD of $(y, \ldots)$-space would be sign-invariant for y and thus the CAD of $(z, y, \ldots)$-space truth invariant for the EC (over admissible cells). But we can no longer say only sections are admissible for the next lift as there may be cells with $z \neq 0$ and $y = 0$. We must instead lift over all cells of $(y, \ldots)$-space, saying:

- Over sections of y: z is no longer an EC (as $zy = 0$ is forced by $y = 0$), so we lift onto all polynomials.
- Over sectors of y: z is an EC, so we only lift with respect to this.

In $(z, y, \ldots)$-space, some cells are admissible (either $y = 0$ or $z = 0$) and the rest are not ($zy \neq 0$). There are difficulties in forming a general algorithm:

1. What would happen if the main variable of the content with respect to x_i were not x_{i-1}?
2. What if the content with respect to x_i were itself not primitive as a polynomial in x_{i-1}?
3. What if there were another equational constraint in x_{i-1}?

The first two can probably (but we have not implemented this yet) be solved by replacing the logic at lines 24– in the algorithm by a dynamic determination of which cells were admissible.

Alternatively in that example we might rewrite ϕ as

$$\phi := (z = 0 \wedge \varphi) \vee (y = 0 \wedge \varphi), \tag{15}$$

so each clause has its own EC. The theory of truth-table invariant CADs (TTICADs) developed by Bradford et al. (2013a, 2016) is designed to deal with such input. More generally, given a formula of the form

$$(f_1 = 0 \wedge g_1 > 0) \vee (f_2 = 0 \wedge g_2 > 0), \tag{16}$$

TTICAD allows for an improvement on the standard EC theory. Since $f_1 f_2 = 0$ is an implicit EC of (16) standard EC theory allows us to avoid studying the g_i away from where any f_i is zero. By utilising a TTICAD from (Bradford et al., 2013a) we can also avoid studying the g_i away from where

the *corresponding* f_i is zero. This approach was extended in (Bradford et al., 2016) to also consider formulae such as

$$(f_1 = 0 \wedge g_1 > 0) \vee (f_2 > 0 \wedge g_2 > 0), \tag{17}$$

where there is no single implicit EC. Although (15) looks closer to (16) it is actually more like (17) since y is not an EC in the main variable.

Although there is the possibility of applying TTICAD for this problem it would first require its own extension to use beyond the first projection (analogous to the present work for standard ECs).

9.2. Classical non primitivity

We can see the importance of the primitivity restriction in the classic complexity results of Brown and Davenport (2007), Davenport and Heintz (1988). Both rest on the following construction. Let $P_k(x_k, y_k)$ be the statement $x_k = f(y_k)$ and then define recursively

$$P_{k-1}(x_{k-1}, y_{k-1}) := \tag{18}$$

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

This is $\exists z_k \left(z_k = f(y_{k-1}) \wedge x_{k-1} = f(z_k)\right)$, i.e. $x_{k-1} = f(f(y_{k-1}))$. Repeated nesting of this procedure builds the doubly-exponential growth. So

$$P_{k-2}(x_{k-2}, y_{k-2}) = Q_{k-1}L_{k-1} \Rightarrow \left(Q_k L_k \Rightarrow P_k(x_k, y_k)\right), \tag{19}$$

gives $x_{k-2} = f(f(f(f(y_{k-2}))))$ etc. Rewriting (19) in prenex form gives

$$P_{k-2}(x_{k-2}, y_{k-2}) = Q_{k-1}Q_k \neg L_{k-1} \vee \neg L_k \vee P_k(x_k, y_k). \tag{20}$$

The negation of (20) is therefore

$$\neg P_{k-2}(x_{k-2}, y_{k-2}) = \overline{Q}_{k-1}\overline{Q}_k L_{k-1} \wedge L_k \wedge \neg P_k(x_k, y_k), \tag{21}$$

where the $\overline{}$ operator interchanges $\forall$ and $\exists$. Now, L_k can be rewritten as

$$\begin{aligned} L_k = (y_{k-1} = y_k \vee y_k = z_k) \wedge (y_{k-1} = y_k \vee x_{k-1} = x_k) \\ \wedge (x_k = z_k \vee y_k = z_k) \wedge (x_k = z_k \vee x_{k-1} = x_k) \end{aligned} \tag{22}$$

and further

$$L_k = (y_{k-1} - y_k)(y_k - z_k) = 0 \wedge (y_{k-1} - y_k)(x_{k-1} - x_k) = 0 \\ \wedge (x_k - z_k)(y_k - z_k) = 0 \wedge (x_k - z_k)(x_{k-1} - x_k) = 0, \qquad (23)$$

which shows L_k to be a conjunction of (non primitive) ECs. This is true for any L_i, hence the propositional part of (21) is a conjunction of eight ECs, mostly non primitive, and $\neg P_k(x_k, y_k)$. Hence by induction we have that the whole family of examples $\neg P_i$ may be written as complete conjunction of (mostly non primitive) ECs. Furthermore there are equalities whose main variables are the first variables to be projected if we try to produce a quantifier-free form of (21). But that quantifier-free form describes the complement of the semi-algebraic varieties in (Brown and Davenport, 2007) or (Davenport and Heintz, 1988) (depending which P_k we take) and these have doubly-exponential complexity in n.

So we observe that the classical results proving the doubly exponential complexity of CAD are not tackled by our EC technology.

10. Lessons for SC^2

The SC^2 community already appreciates that the logical structure of CAD input is important and should be exploited where ever possible. The main additional lesson from the present paper is that this exploitation can take place not only at the Boolean skeleton level but also in the computer algebra.

10.1. Reasons for optimism

There are high barriers to implementing CAD without the support of a computer algebra system, however, SMT solvers such as SMT-RAT by Loup et al. (2013); Kremer and Ábrahám (2019) and Z3 by Jovanovic and de Moura (2012) show it is possible. Indeed, the developers of SMT-RAT are now beginning to expand their CAD module to include a variety of projection operators (Viehmann et al., 2017) and even EC functionality as described by Haehn et al. (2018).

One particular barrier is the need multivariate factorization algorithms, which those developing in a computer algebra system can take for granted but represent a significant implementation cost. On this point we highlight to the SMT community the availability of CoCoALib which is a free C++ library that can perform computer algebra computations without the requirement for

the accompanying Computer Algebra System (in this case CoCoA) (Abbott and Bigatti, 2014). Further, CoCoA is now actively developing features for use in SMT as described by Abbott and Bigatti (2017); Abbott et al. (2018).

10.2. Incrementality

A key requirement for the effective use of CAD by SMT-solvers is that the CAD technology be incremental: that polynomials can be added and removed to the input with the data structures of the CAD edited rather than recalculated. Such incremental CAD algorithms are now under development as part of the SC^2 by Kremer and Ábrahám (2019); Cowen-Rivers and England (2018).

An additional advantage from incremental CAD would be with regards to the issue of well-orientedness. I.e. if a particular operator is found to not be well-oriented at the end of a CAD calculation the next step would be to revert to a less efficient operator which is a superset of the original. Refining an existing decomposition should be cheaper than recomputing from scratch. Although on this point, the development of the Lazard projection theory may remove the well-orientedness condition all together.

However, the use of CAD with ECs incrementally requires additional development work. First, it introduces additional decisions to be taken such as EC designation and whether to pre-processing with GB (not to mention whether that can also be done incrementally). Second, this growing number of decisions needs to be taken in tandem, prompting exponential growth in the number of possibilities that overwhelms existing heuristics. Machine learning techniques may be one way forward, as outlined by England (2018).

Finally, existing heuristics that guide the Boolean search may not be suitable since the use of ECs could prompt what appears as strange behaviour in the SMT context. For example, removing a constraint that was equational could actually grow the output CAD since it necessitates the use of a larger projection operator. Correspondingly, adding an equational constraint could allow a smaller operator and shrink the output. SMT solver search heuristics will need to be adapted to handle these possibilities.

11. Summary

We have presented much of the state of the art in the theory of CAD with Equational Constraints. This included how ECs may be leveraged for savings in the lifting phase as well as projection. We demonstrated the benefits of the theory with worked examples and complexity analysis. The latter shows that the worst CAD bound has double exponent that reduced from n by the number of ECs. Crucially, this is the global double exponent covering both the number and degree of polynomials, if we allow for Groebner Basis pre-processing.

The main avenues for future work are an exploration of dealing with non-primitive ECs; the extension of the Lazard projection operator to a family of operators for ECs; the development of heuristics for choosing which ECs to designate; and the development of incremental EC technology. We note that the current results and any future progress have benefits not only for Symbolic Computation but the wider SC^2 community.

Acknowledgements

This work was originally supported by EPSRC grant EP/J003247/1 and later by EU H2020-FETOPEN-2016-2017-CSA project $\mathcal{SC}^2$ (712689).

We are grateful to all the anonymous referees of this paper, and also those of our conference papers at ISSAC 2015 (England et al., 2015) and CASC 2016 (England and Davenport, 2016), for many helpful comments. We are particularly grateful to the referee who pointed out the mistake in the literature on operator (7) and Dr McCallum for discussing this with us.

We also thank Prof. Buchberger for reminding JHD that Gröbner bases were applicable to the problem of degree growth.

Appendix A. The Iterated Resultants From Section 8.1

$$
\begin{aligned}
R_1 := \operatorname{res}(r_1, r_2, y) = {} & x^{16} + 8\,x^{15} + (-8\,w^2 + 8\,w + 64)x^{14} + (-56\,w^2 + 56\,w \\
& + 288)x^{13} + (28\,w^4 - 56\,w^3 - 332\,w^2 + 400\,w + 1138)x^{12} + (168\,w^4 \\
& - 336\,w^3 - 1144\,w^2 + 1552\,w + 2912)x^{11} + (-56\,w^6 + 168\,w^5 + 648\,w^4 \\
& - 1816\,w^3 - 2664\,w^2 + 5328\,w + 6336)x^{10} + (-280\,w^6 + 840\,w^5 \\
& + 1400\,w^4 - 5400\,w^3 - 2616\,w^2 + 11368\,w + 7808)x^9 + (70\,w^8 \\
& - 280\,w^7 - 500\,w^6 + 3080\,w^5 - 270\,w^4 - 11576\,w^3 + 4860\,w^2 \\
& + 20816\,w + 7381)x^8 + (280\,w^8 - 1120\,w^7 + 80\,w^6 + 6080\,w^5 - 8480\,w^4 \\
& - 11792\,w^3 + 22840\,w^2 + 20192\,w + 920)x^7 + (-56\,w^{10} + 280\,w^9 \\
& - 80\,w^8 - 2160\,w^7 + 4960\,w^6 + 3200\,w^5 - 22608\,w^4 + 2584\,w^3 \\
& + 40840\,w^2 + 16040\,w + 2024)x^6 + (-168\,w^{10} + 840\,w^9 - 1520\,w^8 \\
& - 1360\,w^7 + 12016\,w^6 - 11296\,w^5 - 23368\,w^4 + 30136\,w^3 + 22032\,w^2 \\
& + 624\,w + 736)x^5 + (28\,w^{12} - 168\,w^{11} + 396\,w^{10} + 160\,w^9 - 3690\,w^8 \\
& + 6576\,w^7 + 4520\,w^6 - 24712\,w^5 + 13154\,w^4 + 37456\,w^3 + 1464\,w^2 \\
& - 1568\,w + 5968)x^4 + (56\,w^{12} - 336\,w^{11} + 1192\,w^{10} - 1680\,w^9 \\
& - 2688\,w^8 + 12496\,w^7 - 13464\,w^6 - 16912\,w^5 + 37240\,w^4 + 13472\,w^3 \\
& - 16384\,w^2 + 1984\,w + 3072)x^3 + (-8\,w^{14} + 56\,w^{13} - 248\,w^{12} + 520\,w^{11} \\
& + 72\,w^{10} - 3088\,w^9 + 7664\,w^8 - 2040\,w^7 - 16176\,w^6 + 20424\,w^5 \\
& + 20056\,w^4 - 15360\,w^3 - 8544\,w^2 + 4608\,w + 2304)x^2 + (-8\,w^{14} \\
& + 56\,w^{13} - 296\,w^{12} + 808\,w^{11} - 1144\,w^{10} - 776\,w^9 + 6184\,w^8 - 7048\,w^7 \\
& - 6944\,w^6 + 19696\,w^5 + 3872\,w^4 - 16832\,w^3 - 1152\,w^2 + 4608\,w)x + w^{16} \\
& - 8\,w^{15} + 52\,w^{14} - 184\,w^{13} + 454\,w^{12} - 440\,w^{11} - 772\,w^{10} + 3352\,w^9 \\
& - 2447\,w^8 - 4288\,w^7 + 8200\,w^6 + 2080\,w^5 - 7664\,w^4 - 384\,w^3 + 2304\,w^2 \\
R_2 := \operatorname{res}(r_1, r_3, y) = {} & x^{16} + 8\,x^{15} + (-8\,w^2 + 8\,w + 28)x^{14} + (-56\,w^2 + 56\,w \\
& + 48)x^{13} + (28\,w^4 - 56\,w^3 - 116\,w^2 + 160\,w - 2)x^{12} + (168\,w^4 \\
& - 336\,w^3 + 80\,w^2 + 184\,w - 256)x^{11} + (-56\,w^6 + 168\,w^5 + 108\,w^4 \\
& - 592\,w^3 + 852\,w^2 - 240\,w - 12)x^{10} + (-280\,w^6 + 840\,w^5 - 1120\,w^4 \\
& + 360\,w^3 + 1872\,w^2 - 1448\,w + 2000)x^9 + (70\,w^8 - 280\,w^7 + 220\,w^6 \\
& + 560\,w^5 - 2742\,w^4 + 3232\,w^3 - 1428\,w^2 + 224\,w + 4537)x^8 + (280\,w^8 \\
& - 1120\,w^7 + 2720\,w^6 - 3280\,w^5 - 1280\,w^4 + 6016\,w^3 - 11696\,w^2 + 7496\,w \\
& + 2552)x^7 + (-56\,w^{10} + 280\,w^9 - 620\,w^8 + 480\,w^7 + 2488\,w^6 - 6880\,w^5 \\
& + 9384\,w^4 - 5744\,w^3 - 9404\,w^2 + 12008\,w - 4120)x^6 + (-168\,w^{10}
\end{aligned}
$$

$$
\begin{aligned}
&+ 840\,w^9 - 2960\,w^8 + 5840\,w^7 - 4832\,w^6 - 3088\,w^5 + 21104\,w^4 \\
&- 27128\,w^3 + 12552\,w^2 + 3888\,w - 5888)x^5 + (28\,w^{12} - 168\,w^{11} + 612\,w^{10} \\
&- 1280\,w^9 + 498\,w^8 + 3648\,w^7 - 12424\,w^6 + 17360\,w^5 - 4546\,w^4 \\
&- 13928\,w^3 + 19032\,w^2 - 9344\,w - 176)x^4 + (56w^{12} - 336w^{11} + 1552w^{10} \\
&- 4200\,w^9 + 7296\,w^8 - 6080\,w^7 - 7440\,w^6 + 25880\,w^5 - 31352\,w^4 \\
&+ 13472\,w^3 + 1856\,w^2 - 10304\,w + 1536)x^3 + (-8\,w^{14} + 56\,w^{13} - 284\,w^{12} \\
&+ 880\,w^{11} - 1740\,w^{10} + 1616\,w^9 + 2468\,w^8 - 10704\,w^7 + 15828\,w^6 \\
&- 8040\,w^5 - 1064\,w^4 + 9792\,w^3 - 3168\,w^2 + 2304)x^2 + (-8\,w^{14} + 56\,w^{13} \\
&- 320\,w^{12} + 1096\,w^{11} - 2800\,w^{10} + 4600\,w^9 - 3968\,w^8 - 2152\,w^7 \\
&+ 9592\,w^6 - 10832\,w^5 + 5312\,w^4 + 4672\,w^3 - 5760\,w^2 + 4608\,w)x + w^{16} \\
&- 8\,w^{15} + 52\,w^{14} - 208\,w^{13} + 646\,w^{12} - 1376\,w^{11} + 2012\,w^{10} - 1136\,w^9 \\
&- 1295\,w^8 + 4328\,w^7 - 3992\,w^6 + 2368\,w^5 + 2320\,w^4 - 1920\,w^3 + 2304\,w^2
\end{aligned}
$$

$$
\begin{aligned}
R_3 := \operatorname{res}(r_3, r_3, y) = {} & x^{16} + 8\,x^{15} + (-8\,w^2 + 8\,w + 44)x^{14} + (-56\,w^2 + 56\,w \\
&+ 160)x^{13} + (28\,w^4 - 56\,w^3 - 228\,w^2 + 272\,w + 430)x^{12} + (168\,w^4 \\
&- 336\,w^3 - 592\,w^2 + 856\,w + 816)x^{11} + (-56\,w^6 + 168\,w^5 + 444\,w^4 \\
&- 1264\,w^3 - 812\,w^2 + 1952\,w + 1092)x^{10} + (-280\,w^6 + 840\,w^5 + 560\,w^4 \\
&- 3000\,w^3 + 32\,w^2 + 3032\,w + 736)x^9 + (70\,w^8 - 280\,w^7 - 340\,w^6 \\
&+ 2240\,w^5 - 902\,w^4 - 4208\,w^3 + 2716\,w^2 + 3120\,w - 183)x^8 + (280\,w^8 \\
&- 1120\,w^7 + 480\,w^6 + 3440\,w^5 - 4640\,w^4 - 2304\,w^3 + 5840\,w^2 + 1128\,w \\
&- 1144)x^7 + (-56\,w^{10} + 280\,w^9 - 60\,w^8 - 1760\,w^7 + 3128\,w^6 + 960\,w^5 \\
&- 7352\,w^4 + 3216\,w^3 + 5860\,w^2 - 1320\,w - 824)x^6 + (-168\,w^{10} + 840\,w^9 \\
&- 1280\,w^8 - 880\,w^7 + 5568\,w^6 - 5008\,w^5 - 4464\,w^4 + 7848\,w^3 + 984\,w^2 \\
&- 2576\,w - 64)x^5 + (28\,w^{12} - 168\,w^{11} + 276\,w^{10} + 400\,w^9 - 2302\,w^8 \\
&+ 2848\,w^7 + 1880\,w^6 - 7440\,w^5 + 3582\,w^4 + 5704\,w^3 - 3208\,w^2 - 1216\,w \\
&+ 720)x^4 + (56\,w^{12} - 336\,w^{11} + 880\,w^{10} - 840\,w^9 - 1424\,w^8 + 4800\,w^7 \\
&- 3856\,w^6 - 3464\,w^5 + 6968\,w^4 + 32\,w^3 - 3392\,w^2 + 448\,w + 512)x^3 \\
&+ (-8\,w^{14} + 56\,w^{13} - 172\,w^{12} + 208\,w^{11} + 308\,w^{10} - 1504\,w^9 + 1972\,w^8 \\
&+ 432\,w^7 - 3788\,w^6 + 2920\,w^5 + 2552\,w^4 - 3136\,w^3 - 864\,w^2 + 1024\,w \\
&+ 256)x^2 + (-8\,w^{14} + 56\,w^{13} - 208\,w^{12} + 424\,w^{11} - 352\,w^{10} - 520\,w^9 \\
&+ 1744\,w^8 - 1416\,w^7 - 1176\,w^6 + 2928\,w^5 - 384\,w^4 - 1984\,w^3 + 384\,w^2 \\
&+ 512\,w)x + w^{16} - 8\,w^{15} + 36\,w^{14} - 96\,w^{13} + 150\,w^{12} - 48\,w^{11} - 308\,w^{10} \\
&+ 672\,w^9 - 351\,w^8 - 648\,w^7 + 1096\,w^6 - 880\,w^4 + 128\,w^3 + 256\,w^2
\end{aligned}
$$

References

Abbott, J., Bigatti, A., 2014. What is new in CoCoA? In: Hong, H., Yap, C. (Eds.), Mathematical Software – ICMS 2014. Vol. 8592 of Lecture Notes in Computer Science. Springer Heidelberg, pp. 352–358.
URL https://doi.org/10.1007/978-3-662-44199-2_55

Abbott, J., Bigatti, A., 2017. New in CoCoA-5.2.0 and CoCoALib-0.99550 for SC-Square. In: England, M., Ganesh, V. (Eds.), Proceedings of the 2nd International Workshop on Satisfiability Checking and Symbolic Computation (SC^2 2017). No. 1974 in CEUR Workshop Proceedings.
URL http://ceur-ws.org/Vol-1974/

Abbott, J., Bigatti, A., Palezzato, E., 2018. New in CoCoA-5.2.4 and CoCoALib-0.99570 for SC-Square. In: Bigatti, A., Brain, M. (Eds.), Proceedings of the 3rd Workshop on Satisfiability Checking and Symbolic Computation (SC^2 2018). No. 2189 in CEUR Workshop Proceedings. pp. 88–94.
URL http://ceur-ws.org/Vol-2189/

Ábrahám, E., Abbott, J., Becker, B., Bigatti, A., Brain, M., Buchberger, B., Cimatti, A., Davenport, J., England, M., Fontaine, P., Forrest, S., Griggio, A., Kroening, D., Seiler, W., Sturm, T., 2016. SC^2: Satisfiability checking meets symbolic computation. In: Kohlhase, M., Johansson, M., Miller, B., de Moura, L., Tompa, F. (Eds.), Intelligent Computer Mathematics: Proceedings CICM 2016. Vol. 9791 of Lecture Notes in Computer Science. Springer International Publishing, pp. 28–43.
URL https://doi.org/10.1007/978-3-319-42547-4_3

Barrett, C., Fontaine, P., Tinelli, C., 2016. The Satisfiability Modulo Theories Library (SMT-LIB). Online Resource. URL http://www.SMT-LIB.org.

Barrett, C., Sebastiani, R., Seshia, S., Tinelli, C., 2009. Satisfiability modulo theories. In: Biere, A., Heule, M., van Maaren, H., Walsh, T. (Eds.), Handbook of Satisfiability (Volume 185 Frontiers in Artificial Intelligence and Applications), Chapter 26. IOS Press, pp. 825–885.

Basu, S., Pollack, R., Roy, M., 2006. Algorithms in Real Algebraic Geometry. Volume 10 of Algorithms and Computations in Mathematics. Springer-Verlag.

Biere, A., Heule, M., van Maaren, H., Walsh, T., 2009. Handbook of Satisfiability (Volume 185 Frontiers in Artificial Intelligence and Applications). IOS Press.

Bradford, R., Chen, C., Davenport, J., England, M., Moreno Maza, M., Wilson, D., 2014. Truth table invariant cylindrical algebraic decomposition by regular chains. In: Gerdt, V., Koepf, W., Seiler, W., Vorozhtsov, E. (Eds.), Computer Algebra in Scientific Computing. Vol. 8660 of Lecture Notes in Computer Science. Springer International Publishing, pp. 44–58.
URL http://dx.doi.org/10.1007/978-3-319-10515-4_4

Bradford, R., Davenport, J., England, M., Errami, H., Gerdt, V., Grigoriev, D., Hoyt, C., Košta, M., Radulescu, O., Sturm, T., Weber, A., 2017. A case study on the parametric occurrence of multiple steady states. In: Proceedings of the 2017 ACM International

Symposium on Symbolic and Algebraic Computation. ISSAC '17. ACM, pp. 45–52.
URL https://doi.org/10.1145/3087604.3087622

Bradford, R., Davenport, J., England, M., McCallum, S., Wilson, D., 2013a. Cylindrical algebraic decompositions for boolean combinations. In: Proceedings of the 38th International Symposium on Symbolic and Algebraic Computation. ISSAC '13. ACM, pp. 125–132.
URL http://dx.doi.org/10.1145/2465506.2465516

Bradford, R., Davenport, J., England, M., McCallum, S., Wilson, D., 2016. Truth table invariant cylindrical algebraic decomposition. Journal of Symbolic Computation 76, 1–35.
URL http://dx.doi.org/10.1016/j.jsc.2015.11.002

Bradford, R., Davenport, J., England, M., Wilson, D., 2013b. Optimising problem formulations for cylindrical algebraic decomposition. In: Carette, J., Aspinall, D., Lange, C., Sojka, P., Windsteiger, W. (Eds.), Intelligent Computer Mathematics. Vol. 7961 of Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 19–34.
URL http://dx.doi.org/10.1007/978-3-642-39320-4_2

Brown, C., 2001. Improved projection for cylindrical algebraic decomposition. Journal of Symbolic Computation 32 (5), 447–465.
URL https://doi.org/10.1006/jsco.2001.0463

Brown, C., 2003. QEPCAD B: A program for computing with semi-algebraic sets using CADs. ACM SIGSAM Bulletin 37 (4), 97–108.
URL https://doi.org/10.1145/968708.968710

Brown, C., 2005. The McCallum projection, lifting, and order-invariance. Tech. rep., U.S. Naval Academy, Computer Science Department.
URL https://www.usna.edu/Users/cs/cstech/tr/reports/2005-02.orig.pdf

Brown, C., 2013. Constructing a single open cell in a cylindrical algebraic decomposition. In: Proceedings of the 38th International Symposium on Symbolic and Algebraic Computation. ISSAC '13. ACM, pp. 133–140.
URL https://doi.org/10.1145/2465506.2465952

Brown, C., 2015. Open non-uniform cylindrical algebraic decompositions. In: Proceedings of the 2015 International Symposium on Symbolic and Algebraic Computation. ISSAC '15. ACM, pp. 85–92.
URL https://doi.org/10.1145/2755996.2756654

Brown, C., Davenport, J., 2007. The complexity of quantifier elimination and cylindrical algebraic decomposition. In: Proceedings of the 2007 International Symposium on Symbolic and Algebraic Computation. ISSAC '07. ACM, pp. 54–60.
URL https://doi.org/10.1145/1277548.1277557

Brown, C., Kahoui, M. E., Novotni, D., Weber, A., 2006. Algorithmic methods for investigating equilibria in epidemic modeling. Journal of Symbolic Computation 41, 1157–1173.
URL https://doi.org/10.1016/j.jsc.2005.09.011

Brown, C., McCallum, S., 2005. On using bi-equational constraints in CAD construction. In: Proceedings of the 2005 International Symposium on Symbolic and Algebraic Computation. ISSAC '05. ACM, pp. 76–83.
URL https://doi.org/10.1145/1073884.1073897

Buchberger, B., 2006. 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.
URL https://doi.org/10.1016/j.jsc.2005.09.007

Buchberger, B., Hong, H., 1991. Speeding up quantifier elimination by Gröbner bases. Tech. rep., 91-06. RISC, Johannes Kepler University.
URL http://www3.risc.jku.at/publications/download/risc_1875/1991-02-08-A.pdf

Busé, L., Mourrain, B., 2009. Explicit factors of some iterated resultants and discriminants. Mathematics of Computation 78, 345–386.
URL https://doi.org/10.1090/S0025-5718-08-02111-X

Caviness, B., Johnson, J., 1998. Quantifier Elimination and Cylindrical Algebraic Decomposition. Texts & Monographs in Symbolic Computation. Springer-Verlag.

Charalampakis, A., Chatzigiannelis, I., 2018. Analytical solutions for the minimum weight design of trusses by cylindrical algebraic decomposition. Archive of Applied Mechanics 88 (1), 39–49.
URL https://doi.org/10.1007/s00419-017-1271-8

Chen, C., Moreno Maza, M., Xia, B., Yang, L., 2009. Computing cylindrical algebraic decomposition via triangular decomposition. In: Proceedings of the 2009 International Symposium on Symbolic and Algebraic Computation. ISSAC '09. ACM, pp. 95–102.
URL https://doi.org/10.1145/1576702.1576718

Collins, G., 1975. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In: Proceedings of the 2nd GI Conference on Automata Theory and Formal Languages. Springer-Verlag (reprinted in the collection Caviness and Johnson (1998)), pp. 134–183.
URL https://doi.org/10.1007/3-540-07407-4_17

Collins, G., 1998. Quantifier elimination by cylindrical algebraic decomposition – 20 years of progress. In: Caviness, B., Johnson, J. (Eds.), Quantifier Elimination and Cylindrical Algebraic Decomposition. Texts & Monographs in Symbolic Computation. Springer-Verlag, pp. 8–23.
URL https://doi.org/10.1007/978-3-7091-9459-1_2

Collins, G., Hong, H., 1991. Partial cylindrical algebraic decomposition for quantifier elimination. Journal of Symbolic Computation 12, 299–328.

Cowen-Rivers, A., England, M., 2018. Towards incremental cylindrical algebraic decomposition in Maple. In: Bigatti, A., Brain, M. (Eds.), Proceedings of the 3rd Workshop on Satisfiability Checking and Symbolic Computation (SC^2 2018). No. 2189 in CEUR Workshop Proceedings. pp. 3–18.
URL http://ceur-ws.org/Vol-2189/

Davenport, J., Bradford, R., England, M., Wilson, D., 2012. Program verification in the presence of complex numbers, functions with branch cuts etc. In: 14th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing. SYNASC '12. IEEE, pp. 83–88.
URL http://dx.doi.org/10.1109/SYNASC.2012.68

Davenport, J., Heintz, J., 1988. Real quantifier elimination is doubly exponential. Journal

of Symbolic Computation 5 (1-2), 29–35.
URL https://doi.org/10.1016/S0747-7171(88)80004-X

Dolzmann, A., Seidl, A., Sturm, T., 2004. Efficient projection orders for CAD. In: Proceedings of the 2004 International Symposium on Symbolic and Algebraic Computation. ISSAC '04. ACM, pp. 111–118.
URL https://doi.org/10.1145/1005285.1005303

England, M., 2018. Machine learning for mathematical software. In: Davenport, J., Kauers, M., Labahn, G., Urban, J. (Eds.), Mathematical Software – Proc. ICMS 2018. Vol. 10931 of Lecture Notes in Computer Science. Springer International Publishing, pp. 165–174.
URL https://doi.org/10.1007/978-3-319-96418-8_20

England, M., Bradford, R., Chen, C., Davenport, J., Moreno Maza, M., Wilson, D., 2014a. Problem formulation for truth-table invariant cylindrical algebraic decomposition by incremental triangular decomposition. In: Watt, S., Davenport, J., Sexton, A., Sojka, P., Urban, J. (Eds.), Intelligent Computer Mathematics. Vol. 8543 of Lecture Notes in Artificial Intelligence. Springer International, pp. 45–60.
URL http://dx.doi.org/10.1007/978-3-319-08434-3_5

England, M., Bradford, R., Davenport, J., 2015. Improving the use of equational constraints in cylindrical algebraic decomposition. In: Proceedings of the 2015 International Symposium on Symbolic and Algebraic Computation. ISSAC '15. ACM, pp. 165–172.
URL http://dx.doi.org/10.1145/2755996.2756678

England, M., Davenport, J., 2016. The complexity of cylindrical algebraic decomposition with respect to polynomial degree. In: Gerdt, V., Koepf, W., Werner, W., Vorozhtsov, E. (Eds.), Computer Algebra in Scientific Computing: 18th International Workshop, CASC 2016. Vol. 9890 of Lecture Notes in Computer Science. Springer International Publishing, pp. 172–192.
URL http://dx.doi.org/10.1007/978-3-319-45641-6_12

England, M., Errami, H., Grigoriev, D., Radulescu, O., Sturm, T., Weber, A., 2017. Symbolic versus numerical computation and visualization of parameter regions for multistationarity of biological networks. In: Gerdt, V., Koepf, W., Seiler, W., Vorozhtsov, E. (Eds.), Computer Algebra in Scientific Computing (CASC). Vol. 10490 of Lecture Notes in Computer Science. Springer International Publishing, pp. 93–108.
URL https://doi.org/10.1007/978-3-319-66320-3_8

England, M., Wilson, D., Bradford, R., Davenport, J., 2014b. Using the Regular Chains Library to build cylindrical algebraic decompositions by projecting and lifting. In: Hong, H., Yap, C. (Eds.), Mathematical Software – ICMS 2014. Vol. 8592 of Lecture Notes in Computer Science. Springer Heidelberg, pp. 458–465.
URL http://dx.doi.org/10.1007/978-3-662-44199-2_69

Erascu, M., Hong, H., 2016. Real quantifier elimination for the synthesis of optimal numerical algorithms (Case study: Square root computation). Journal of Symbolic Computation 75, 110–126.
URL https://doi.org/10.1016/j.jsc.2015.11.010

Faugère, J., 2002. A new efficient algorithm for computing Gröbner bases without reduction to zero (F5). In: Proceedings of the 2002 International Symposium on Symbolic

and Algebraic Computation. ISSAC '02. ACM, pp. 75–83.
URL https://doi.org/10.1145/780506.780516

Fotiou, I., Parrilo, P., Morari, M., 2005. Nonlinear parametric optimization using cylindrical algebraic decomposition. In: Decision and Control, 2005 European Control Conference. CDC-ECC '05. pp. 3735–3740.
URL https://doi.org/10.1109/CDC.2005.1582743

Fukasaku, R., Iwane, H., Sato, Y., 2015. Real quantifier elimination by computation of comprehensive Gröbner systems. In: Proceedings of the 2015 International Symposium on Symbolic and Algebraic Computation. ISSAC '15. ACM, pp. 173–180.
URL https://doi.org/10.1145/2755996.2756646

Haehn, R., Kremer, G., Ábrahám, E., 2018. Evaluation of equational constraints for CAD in SMT solving. In: Bigatti, A., Brain, M. (Eds.), Proceedings of the 3rd Workshop on Satisfiability Checking and Symbolic Computation (SC^2 2018). No. 2189 in CEUR Workshop Proceedings. pp. 19–32.
URL http://ceur-ws.org/Vol-2189/

Han, J., Dai, L., Xia, B., 2014. Constructing fewer open cells by gcd computation in CAD projection. In: Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation. ISSAC '14. ACM, pp. 240–247.
URL https://doi.org/10.1145/2608628.2608676

Hong, H., 1990. An improvement of the projection operator in cylindrical algebraic decomposition. In: Proceedings of the International Symposium on Symbolic and Algebraic Computation. ISSAC '90. ACM, pp. 261–264.
URL https://doi.org/10.1145/96877.96943

Huang, Z., England, M., Davenport, J., Paulson, L., 2016. Using machine learning to decide when to precondition cylindrical algebraic decomposition with Groebner bases. In: 18th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC '16). IEEE, pp. 45–52.
URL https://doi.org/10.1109/SYNASC.2016.020

Huang, Z., England, M., Wilson, D., Davenport, J., Paulson, L., Bridge, J., 2014. Applying machine learning to the problem of choosing a heuristic to select the variable ordering for cylindrical algebraic decomposition. In: Watt, S., Davenport, J., Sexton, A., Sojka, P., Urban, J. (Eds.), Intelligent Computer Mathematics. Vol. 8543 of Lecture Notes in Artificial Intelligence. Springer International, pp. 92–107.
URL http://dx.doi.org/10.1007/978-3-319-08434-3_8

Iwane, H., Yanami, H., Anai, H., Yokoyama, K., 2009. An effective implementation of a symbolic-numeric cylindrical algebraic decomposition for quantifier elimination. In: Proceedings of the 2009 conference on Symbolic Numeric Computation. SNC '09. pp. 55–64.
URL https://doi.org/10.1145/1577190.1577203

Jouanolou, J., 1991. Le formalisme du résultant. Advances in Mathematics 90 (2), 117–263.
URL https://doi.org/10.1016/0001-8708(91)90031-2

Jovanovic, D., de Moura, L., 2012. Solving non-linear arithmetic. In: Gramlich, B., Miller, D., Sattler, U. (Eds.), Automated Reasoning: 6th International Joint Conference (IJ-

CAR). Vol. 7364 of Lecture Notes in Computer Science. Springer, pp. 339–354.
URL https://doi.org/10.1007/978-3-642-31365-3_27

Kremer, G., Ábrahám, E., 2019. Fully incremental CAD. In Press.

Kroening, D., Strichman, O., 2013. Decision Procedures: An Algorithmic Point of View. Springer, New York.

Lazard, D., 1994. An improved projection for cylindrical algebraic decomposition. In: Bajaj, C. (Ed.), Algebraic Geometry and its Applications: Collections of Papers from Abhyankar's 60th Birthday Conference. Springer Berlin, pp. 467–476.
URL https://doi.org/10.1007/978-1-4612-2628-4_29

Lazard, D., McCallum, S., 2009. Iterated discriminants. Journal of Symbolic Computation 44 (9), 1176–1193.
URL https://doi.org/10.1016/j.jsc.2008.05.006

Loup, U., Scheibler, K., Corzilius, F., Ábrahám, E., Becker, B., 2013. A symbiosis of interval constraint propagation and cylindrical algebraic decomposition. In: Bonacina, M. (Ed.), Automated Deduction (CADE-24). Vol. 7898 of Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 193–207.
URL https://doi.org/10.1007/978-3-642-38574-2_13

Mayr, E., Meyer, A., 1982. The complexity of the word problems for commutative semigroups and polynomial ideals. Advances in Mathematics 46 (3), 305–329.
URL https://doi.org/10.1016/0001-8708(82)90048-2

Mayr, E., Ritscher, S., 2013. Dimension-dependent bounds for Gröbner bases of polynomial ideals. Journal of Symbolic Computation 49, 78–94.
URL https://doi.org/10.1016/j.jsc.2011.12.018

McCallum, S., 1985. An improved projection operation for cylindrical algebraic decomposition. PhD Thesis (Computer Sciences Technical Report 578), University of Wisconsin-Madison.

McCallum, S., 1998. An improved projection operation for cylindrical algebraic decomposition. In: Caviness, B., Johnson, J. (Eds.), Quantifier Elimination and Cylindrical Algebraic Decomposition. Texts & Monographs in Symbolic Computation. Springer-Verlag, pp. 242–268.
URL https://doi.org/10.1007/978-3-7091-9459-1_12

McCallum, S., 1999a. Factors of iterated resultants and discriminants. Journal of Symbolic Computation 27 (4), 367–385.
URL https://doi.org/10.1006/jsco.1998.0257

McCallum, S., 1999b. On projection in CAD-based quantifier elimination with equational constraint. In: Proceedings of the 1999 International Symposium on Symbolic and Algebraic Computation. ISSAC '99. ACM, pp. 145–149.
URL https://doi.org/10.1145/309831.309892

McCallum, S., 2001. On propagation of equational constraints in CAD-based quantifier elimination. In: Proceedings of the 2001 International Symposium on Symbolic and Algebraic Computation. ISSAC '01. ACM, pp. 223–231.
URL https://doi.org/10.1145/384101.384132

McCallum, S., Brown, C., 2009. On delineability of varieties in CAD-based quantifier elimination with two equational constraints. In: Proceedings of the 2009 International

Symposium on Symbolic and Algebraic Computation. ISSAC '09. ACM, pp. 71–78.
URL https://doi.org/10.1145/1576702.1576715

McCallum, S., Hong, H., 2016. On using Lazard's projection in CAD construction. Journal of Symbolic Computation 72, 65–81.

McCallum, S., Parusiński, A., Paunescu, L., 2019. Validity proof of Lazard's method for CAD construction. Journal of Symbolic Computation 92, 52–69.
URL https://doi.org/10.1016/j.jsc.2017.12.002

Mulligan, C., Bradford, R., Davenport, J., England, M., Tonks, Z., 2018a. Non-linear real arithmetic benchmarks derived from automated reasoning in economics. In: Bigatti, A., Brain, M. (Eds.), Proceedings of the 3rd Workshop on Satisfiability Checking and Symbolic Computation (SC^2 2018). No. 2189 in CEUR Workshop Proceedings. pp. 48–60.
URL http://ceur-ws.org/Vol-2189/

Mulligan, C., Davenport, J., England, M., 2018b. TheoryGuru: A Mathematica package to apply quantifier elimination technology to economics. In: Davenport, J., Kauers, M., Labahn, G., Urban, J. (Eds.), Mathematical Software – Proc. ICMS 2018. Vol. 10931 of Lecture Notes in Computer Science. Springer International Publishing, pp. 369–378.
URL https://doi.org/10.1007/978-3-319-96418-8_44

Paulson, L., 2012. Metitarski: Past and future. In: Beringer, L., Felty, A. (Eds.), Interactive Theorem Proving. Vol. 7406 of Lecture Notes in Computer Science. Springer, pp. 1–10.
URL https://doi.org/10.1007/978-3-642-32347-8_1

Schwartz, J., Sharir, M., 1983. On the "Piano-Movers" Problem: II. General techniques for computing topological properties of real algebraic manifolds. Adv. Appl. Math. 4, 298–351.
URL https://doi.org/10.1016/0196-8858(83)90014-3

Seidl, A., 2006. Cylindrical decomposition under application-oriented paradigms. Ph.D. thesis, Universitt Passau, Fakultt fr Informatik und Mathematik.
URL https://opus4.kobv.de/opus4-uni-passau/files/46/Seidl_Andreas.pdf

Strzeboński, A., 2006. Cylindrical algebraic decomposition using validated numerics. Journal of Symbolic Computation 41 (9), 1021–1038.
URL https://doi.org/10.1016/j.jsc.2006.06.004

Strzeboński, A., 2016. Cylindrical algebraic decomposition using local projections. Journal of Symbolic Computation 76, 36–64.
URL https://doi.org/10.1016/j.jsc.2015.11.018

Viehmann, T., Kremer, G., Ábráham, E., 2017. Comparing different projection operators in the cylindrical algebraic decomposition for smt solving. In: England, M., Ganesh, V. (Eds.), Proceedings of the 2nd International Workshop on Satisfiability Checking and Symbolic Computation (SC^2 2017). No. 1974 in CEUR Workshop Proceedings.
URL http://ceur-ws.org/Vol-1974/

Wada, Y., Matsuzaki, T., Terui, A., Arai, N., 2016. An automated deduction and its implementation for solving problem of sequence at university entrance examination. In: Greuel, G.-M., Koch, T., Paule, P., Sommese, A. (Eds.), Mathematical Software – Proceedings of ICMS 2016. Vol. 9725 of Lecture Notes in Computer Science. Springer

International Publishing, pp. 82–89.
URL https://doi.org/10.1007/978-3-319-42432-3_11

Wilson, D., Bradford, R., Davenport, J., 2012. Speeding up cylindrical algebraic decomposition by Gröbner bases. In: Jeuring, J., Campbell, J., Carette, J., Reis, G., Sojka, P., Wenzel, M., Sorge, V. (Eds.), Intelligent Computer Mathematics. Vol. 7362 of Lecture Notes in Computer Science. Springer, pp. 280–294.
URL https://doi.org/10.1007/978-3-642-31374-5_19

Wilson, D., England, M., Davenport, J., Bradford, R., 2014. Using the distribution of cells by dimension in a cylindrical algebraic decomposition. In: 16th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing. SYNASC '14. IEEE, pp. 53–60.
URL http://dx.doi.org/10.1109/SYNASC.2014.15