Curtains in CAD: Why are they a problem and how do we fix them?*

Akshar Nair[1 [0000−0001−7379−1868]], James Davenport[1 [0000−0002−3982−7545]], and Gregory Sankaran[1 [0000−0002−5846−6490]]

University of Bath, Claverton Down, Bath, BA2 7AY, United Kingdom asn42@bath.ac.uk, masjhd@bath.ac.uk, masgks@bath.ac.uk

Abstract. In our earlier work [11], we combined the CAD methods of McCallum [5] and Lazard [4] so as to produce an efficient algorithm for decomposing a hypersurface rather than the whole of $\mathbb{R}^n$ (exploiting an equational constraint $f = 0$). That method, however, fails if f is nullified (in the terminology of [5]): we call the set where this happens a curtain. Here we provide a further modification which, at the cost of a trade off in terms of complexity as compared to [11], is valid for any hypersurface, including one containing curtains.

1 Introduction

A Cylindrical Algebraic Decomposition (CAD) is a decomposition of a semi-algebraic set $S \subseteq \mathbb{R}^n$ (for any n) into semi-algebraic sets (also known as cells) homeomorphic to $\mathbb{R}^m$, where $0 \leq m \leq n$, such that the projection of any two cells onto the first k coordinates is either the same or disjoint. We generally want the cells to have some property relative to some given set of input polynomials, often referred to as constraints. For example, we might require sign-invariance, i.e. the sign of each input polynomial is constant on each cell, as in the original algorithm of [2].

CAD algorithms have many applications: epidemic modelling [1], artificial intelligence to pass exams [12], financial analysis [10], and many more, so efficient algorithms are of more than theoretical interest.

If S is contained in a subvariety of $\mathbb{R}^n$ it is clearly wasteful to compute a decomposition of $\mathbb{R}^n$. McCallum, in [6], adapts his earlier algorithm [5] to this this situation. To explain this idea more precisely, we need some terminology.

Definition 1. *A Quantifier Free Tarski Formula (QFF) is made up of atoms connected by the standard boolean operators $\wedge, \vee$ and $\neg$. The atoms are statements about signs of polynomials $f \in \mathbb{R}[x_1, \ldots, x_n]$, of the form $f * 0$ where $* \in \{=, <, >\}$ (and by combination also $\{\geq, \leq, \neq\}$).*

Strictly speaking we need only the relation $<$, but this form is more convenient because of the next definition.

Definition 2. *[3] 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*; if not, then it is* implicit*. If the constraint is visibly an equality one from the formula, i.e. the formula Φ is $f = 0 \wedge \Phi'$, we say the constraint is* syntactically explicit*.*

* Supported by University of Bath and EPSRC

Although implicit and explicit ECs have the same logical status, in practice only the syntactically explicit ECs will be known to us and therefore be available to be exploited.

Example 1. [3] Let f and g be two real polynomials.

1. The formula $f = 0 \wedge g > 0$ has an explicit EC, $f = 0$.
2. The formula $f = 0 \vee g = 0$ has no explicit EC, but the equation $fg = 0$ is an implicit EC.
3. The formula $f^2 + g^2 \leq 0$ also has no explicit EC, but it has two implicit ECs: $f = 0$ and $g = 0$.
4. The formula $f = 0 \vee f^2 + g^2 \leq 0$ logically implies $f = 0$, and the equation is an atom of the formula which makes it an explicit EC according to the definition. However, since this deduction is semantic rather than syntactic, it is more like an implicit EC rather than an explicit EC.

Definition 3. *Let A be a set of polynomials in $\mathbb{R}[x_1, \ldots, x_n]$ and $P\colon \mathbb{R}[x_1, \ldots, x_n] \times \mathbb{R}^n \to \Sigma$ a function to some set Σ. If $C \subset \mathbb{R}^n$ is a cell and $P(f, \alpha)$ is independent of $\alpha \in C$ for every $f \in A$, then A is called P-invariant over that cell. If this is true for all the cells of a decomposition, we say the decomposition is P-invariant.*

Much work has been done on sign-invariant CADs, but we first focus our attention on the algorithm for lex-least invariant CADs introduced by Lazard [4] (for a validity proof, see [8]). Unlike the algorithm in [5] it works in the presence of curtains (see Definition 10 below), and has some complexity advantages also. In [11] we adapted [4] to the case of an EC, analogously to the adaptation of [5] in [6], but in doing so we reintroduced the problem of curtains. This paper revisits [**?**] and in provides a hybrid algorithm, which we believe to be the first one that gives a CAD of the variety defined by an EC rather than of $\mathbb{R}^n$, and yet is valid even on curtains.

In Section 3 we analyse the reasons why curtains are a problem, and explain how the previous literature has concentrated on valuations rather than curtains themselves. Section 4 consists of the complexity analysis of our algorithm. As in [11], this algorithm cannot be used recursively because the projection operator used in the first stage of projection would output a partial CAD which is a hybrid between sign-invariant and lex-least invariant on curtains of the equational constraint.

2 Lex-least valuation and its applications in CAD

In order to understand lex-least valuation, let us recall *lexicographic order* $\geq_{\text{lex}}$ on $\mathbb{N}^n$, where $n \geq 1$.

Definition 4. *We say that $v = (v_1, \ldots, v_n) \geq_{\text{lex}} (w_1, \ldots, w_n) = w$ if and only if either $v = w$ or there exists an $i \leq n$ such that $v_i > w_i$ and $v_k = w_k$ for all k in the range $1 \leq k < i$.*

Definition 5. *[9, Definition 2.4] Let $n \geq 1$ and suppose that $f \in \mathbb{R}[x_1, \ldots, x_n]$ is non-zero and $\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{R}^n$. The lex-least valuation $\nu_\alpha(f)$ at α is the least (with respect to $\geq_{\text{lex}}$) element $v = (v_1, \ldots, v_n) \in \mathbb{N}^n$ such that f expanded about α has the term*

$$c(x_1 - \alpha_1)^{v_1} \cdots (x_n - \alpha_n)^{v_n},$$

where $c \neq 0$.

Note that $\nu_\alpha(f) = (0, \ldots, 0)$ if and only if $f(\alpha) \neq 0$. The lex-least valuation is referred to as the Lazard valuation in [9].

Example 2. If $n = 1$ and $f(x_1) = x_1^3 - 2x_1^2 + x_1$, then $\nu_0(f) = 1$ and $\nu_1(f) = 2$. If $n = 2$ and $f(x_1, x_2) = x_1(x_2 - 1)^2$, then $\nu_{(0,0)}(f) = (1, 0)$, $\nu_{(2,1)}(f) = (0, 2)$ and $\nu_{(0,1)}(f) = (1, 2)$.

Definition 6. *[9] Let $n \geq 2$, and suppose that $f \in \mathbb{R}[x_1, \ldots, x_n]$ is non-zero and that $\beta \in \mathbb{R}^{n-1}$. The Lazard residue $f_\beta \in \mathbb{R}[x_n]$ of f at β, and the lex-least valuation $N_\beta(f) = (\nu, \ldots, \nu_{n-1})$ of f above β are defined to be the result of Algorithm 1.*

Algorithm 1 Lazard residue

Input: $f \in \mathbb{R}[x_1, \ldots, x_n]$ and $\beta \in \mathbb{R}^{n-1}$.
Output: Lazard residue f_β and Lex-least valuation of f above β.

1: $f_\beta \leftarrow f$
2: **for** $i \leftarrow 1$ to $n - 1$ **do**
3: $\quad \nu_i \leftarrow$ greatest integer ν such that $(x_i - \beta_i)^\nu | f_\beta$.
4: $\quad f_\beta \leftarrow f_\beta / (x_i - \beta_i)^{\nu_i}$.
5: $\quad f_\beta \leftarrow f_\beta(\beta_i, x_{i+1}, \ldots, x_n)$
6: **end for**
7: **return** f_β, $(\nu_1, \ldots, \nu_{n-1})$

Do not confuse the lex-least valuation $N_\beta(f) \in \mathbb{Z}^{n-1}$ of f above $\beta \in \mathbb{R}^{n-1}$ with the lex-least valuation $\nu_\alpha(f) \in \mathbb{Z}^n$ at $\alpha \in \mathbb{R}^n$, defined in Definition 5. Notice that if $\alpha = (\beta, b_n) \in \mathbb{R}^n$ then $\nu_\alpha(f) = (N_\beta(f), \nu_n)$ for some integer ν_n: in other words, $N_\beta(f)$ consists of the first $n-1$ coordinates of the valuation of f at any point above β.

Definition 7. *[9, Definition 2.10] Let $S \subseteq \mathbb{R}^{n-1}$ and $f \in \mathbb{R}[x_1, \ldots, x_n]$. We say that$f$ is Lazard delineable on S if:*

i) The lex-least valuation of f above β is the same for each point $\beta \in S$.
ii) There exist finitely many continuous functions $\theta_i \colon S \to \mathbb{R}$, such that $\theta_1 < \ldots < \theta_k$ if $k > 0$ and such that for all $\beta \in S$, the set of real roots of f_β is $\{\theta_1(\beta), \ldots, \theta_k(\beta)\}$.
iii) If $k = 0$, then the hypersurface $f = 0$ does not pass over S. If $k \geq 1$, then there exist positive integers $m_1, \ldots, m_k$ such that, for all $\beta \in S$ and for all $1 \leq i \leq k$, m_i is the multiplicity of $\theta_i(\beta)$ as a root of f_β.

Definition 8. *[9, Definition 2.10] Let f be Lazard delineable on $S \subseteq \mathbb{R}^{n-1}$. Then*

i) The graphs θ_i are called Lazard sections and m_i is the associated multiplicity of these sections.
ii) The regions between consecutive Lazard sections[1] are called Lazard sectors.

Remark 1. If f is Lazard delineable on S and the valuation of f above any point in S is the zero vector, then the Lazard sections of f are the same as the sections of f defined as in [2] and [6].

Remark 2. We can use Algorithm 1 to compute the lex-least valuation of f at $\alpha \in \mathbb{R}^n$. After the loop is finished, we proceed to the first step of the loop and perform it for $i = n$ and the n-tuple $\nu_1, \ldots, \nu_n$ is the required valuation.

[1] Including $\theta_0 = -\infty$ and $\theta_{k+1} = +\infty$.

Definition 9. *[11] Let $A \subset \mathbb{R}[x_1, \ldots, x_n]$ be a set of polynomials. Let $E \subseteq A$, and define the projection operator $PL_E(A)$ as follows*

$$PL_E(A) = \text{ldcf}(E) \cup \text{trcf}(E) \cup \text{disc}(E) \cup \{\text{res}_{x_n}(f, g) \mid f \in E,\, g \in A \setminus E\}.$$

Here $\text{ldcf}(E)$ and $\text{trcf}(E)$ are the sets of leading and trailing coefficients of elements of E. We will be comparing this to Lazard's projection operator $PL(A)$ defined in [9].

Theorem 1. *[9] Let $f(x, x_n) \in \mathbb{R}[x, x_n]$ have positive degree d in x_n, where $x = (x_1, \ldots, x_{n-1})$. Let $D(x)$, $l(x)$ and $t(x)$ denote the discriminant, leading coefficient and trailing coefficient (with respect to x_n) of f, respectively, and suppose that each of these polynomials is non-zero (as an element of $\mathbb{R}[x]$). Let S be a connected analytic subset of $\mathbb{R}^{n-1}$ in which $D(x)$, $l(x)$ and $t(x)$ are all lex-least invariant. Then f is Lazard delineable on S, and hence f is lex-least invariant in every Lazard section and sector over S. Moreover, the same conclusion holds for the polynomial $f^*(x, x_n) = x_n f(x, x_n)$.*

Remark 3. In practice we will choose E to consist of polynomials occurring in equational constraints.

Theorem 2. *[11] Let $n \geq 2$ and let f, $g \in \mathbb{R}[x_1, \ldots, x_n]$ be of positive degrees in the main variable x_n. Suppose that f is Lazard delineable on a connected subset $S \subset \mathbb{R}^{n-1}$, in which $R = \text{res}_{x_n}(f, g)$ is lex-least invariant, and f does not have a curtain on S (see Definition 11 below). Then g is sign-invariant in each section of f over S.*

3 Implications of curtains on CAD

We propose some geometric terminology to describe the conditions under which a polynomial is nullified in the terminology of [6].

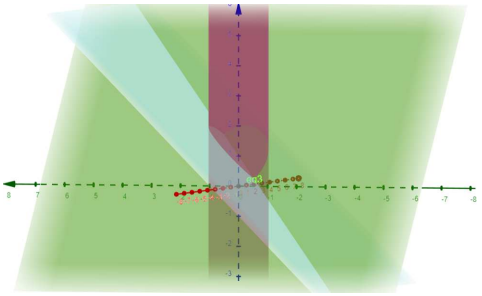
Definition 10. *A variety $C \subseteq \mathbb{R}^n$ is called a curtain if, whenever $(x, x_n) \in C$, then $(x, y) \in C$ for all $y \in \mathbb{R}$.*

Definition 11. *Suppose $f \in \mathbb{R}[x_1, \ldots, x_n]$ and $W \subseteq \mathbb{R}^{n-1}$. We say that f has a curtain at W if for all $\underline{x} \in W$ and $y \in \mathbb{R}$ we have $f(\underline{x}, y) = 0$.*

Remark 4. Lazard delineability differs from delineability as in [2] and [6] in two important ways. First, we require lex-least invariance on the sections. Second, delineability is not defined on curtains, but Lazard delineability is because the Lazard sections are of f_β rather than f.

The algorithms in [6] and [7] exploit ECs but both rely on order-invariant. Because order is not defined on curtains, these algorithms fail there. It is therefore natural to try to use Lazard's algorithm, which does not have this issue when used to construct a full CAD of $\mathbb{R}^n$. However, when we try to exploit an EC, a difficulty does still arise if the EC has a curtain.

Example 3. Let $f = x^2 + y^2 - 1$ (which we assume to be an EC), $g_1 = z - x - 1$ and $g_2 = z - y - 1$ (which we assume are not ECs). Then $\text{res}(f, g_1) = x^2 + y^2 - 1 = \text{res}(f, g_2)$, and this gives us no information about $\text{res}(g_1, g_2)$. In such cases, when the EC has a curtain, it becomes impossible to use PL_E to detect the intersections of the other constraints on that curtain. Because of this, we must resort to a general-purpose projection such as the one in [2], which includes these resultants, when we are on a curtain contained in the hypersurface defined by the EC.



In one common case we can avoid this complication.

Definition 12. *We say that* $f \in \mathbb{R}[x_1, \ldots, x_n]$ *has a point curtain at* $\alpha \in \mathbb{R}^{n-1}$ *if* $N_\alpha(f) \neq (0, \ldots, 0)$ *and there exists a neighbourhood* U *of* α *such that* $N_{\alpha'}(f) = (0, \ldots, 0)$ *for all* $\alpha' \in U \setminus \{\alpha\}$.

In this case we do not need to consider the resultants between the non-equational constraints $A \setminus \{f\}$ when projecting.

Theorem 3. *Let* $f \in \mathbb{R}[x_1, \ldots, x_n]$ *and let* $\alpha \in \mathbb{R}^{n-1}$. *If* f *is an equational constraint and has a point curtain at* α, *then* PL_E *is sufficient to obtain a sign-invariant CAD.*

Indeed, if f has a curtain on a set S of positive dimension, we need the resultants of the non-equational constraints to determine sample points in $\mathbb{R}^{n-1}$ above which two such constraints meet, as in Example 3. For a point curtain, there is only one sample point in $\mathbb{R}^{n-1}$, namely α. We calculate the roots of Lazard residues of all constraints to determine the sample points on the curtain (which is the fibre above α), and nothing more is needed: the algorithm still produces a lex-least invariant CAD.

For this to be useful we need to be able to detect and classify curtains (i.e. tell whether or not they are point curtains). The following describes an algorithm to do this.

Algorithm 2 Detecting and Classifying Curtains

$(B) \leftarrow PC(f, I, A, n)$
Input = Set of indices I, set of sample points S with respect to the indices I, equational constraint $f \in \mathbb{R}[x_1, \ldots, x_{n+1}]$.
Output = B, B', where B is the set of sample points that are point curtains and B' is the set of sample points that are curtains (but not point curtains).

1: $B \leftarrow$ Empty List
2: $B' \leftarrow$ Empty List
3: **for** $\alpha \in A$ **do**
4: **if** $\nu'_\alpha(f) \neq 0$ **then**
5: Check if the nearest 1-cell neighbours have zero valuation.
6: If all neighbours are zero valuation add α to B otherwise add it to B'
7: **end if**
8: **end for**
9: **return** (B,B')

Algorithm 3 Partial CAD for Curtains

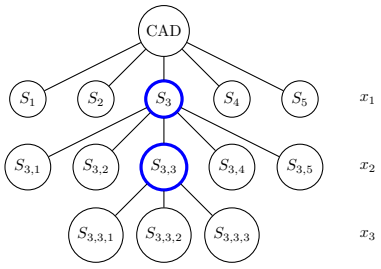
$(I', S') \leftarrow DCBS(A, I, S, C, n)$

Input = Set of indices I, set of sample points S with respect to the indices I, set of polynomials A, C is a list of tuples of sample points that are curtains and their respective indices and n the dimension of our space.

Output = I', S' list of sample points and their indices for sections of the equational constraints that are curtains.

1: If $n \geq 2$ then go to step 3.
2: **for** each $(c, i) \in C$ **do**
3: Isolate the Real roots of the irreducible factors of the non-zero elements of A within the limits of neighbours from c. Construct cell indices I' and sample points S' from the real roots. Exit.
4: **end for**
5: $B \leftarrow$ the square free basis of the primitive parts of all elements of A.
6: $P \leftarrow \text{cont}(A) \cup PL(B)$.
7: $(I'', S'') \leftarrow DBCS(P, I, S, C, n)$.
8: $(I', S') \leftarrow$ (empty list,empty list).
9: **for** each $\alpha \in S''$ **do**
10: Let i be the index of the cell containing α.
11: $f^* \leftarrow \{f_\alpha \mid f \in B\}$.
12: **for** each $(c, i) \in C$ **do**
13: Isolate the real roots of all the polynomials in f^* within the neighbours of c(looking at the nth coordinate).
14: Construct cell indices and sample points for Lazard sections and sectors of elements of B from i, α and the isolated real roots of f^*.
15: Add the sample points to I and S.
16: **end for**
17: **return** (I', S').
18: **end for**

Example 4. Suppose that the list of sample points obtained when projecting using PL_E for the first stage and PL further to that is described by the diagram. If a curtain is detected over the cell described by sample point $S_{3,3}$, then we do the following. First we project all non-equational constraints using Lazard's original projection operator. When computing sample points, we only consider the sample points which lie within the neighbours of the curtain sample point. For example in our case, at level x_1, we calculate all sample points between S_2 and S_4. We lift these sample points to level x_2. Now when computing sample points at level x_2, we consider the points between $S_{3,2}$ and $S_{3,4}$. If at any level the index of the sample point is even, we do not need to calculate roots,but just consider that value as our sample point for lifting.



4 Complexity Analysis of Curtain solving method

In this section we look at the complexity of the number of cells produced by Algorithm 3. Note that Algorithm 3 works so to speak on top of the algorithm described in [11]. The idea is to do a second decomposition on the sections of the equational constraint that contain curtains.

Theorem 4. *Given a set A of m polynomials in n variables with maximum total degree d, Algorithm 3 outputs a partial CAD(i.e. a CAD of $\{f = 0\}$ only) with the number of cells being at most*

$$2^{2^n-1}(m(3m+1)^{2^{n-1}-1} + (m-1)m^{2^{n-1}-1})d^{2^n-1}. \tag{1}$$

This complexity is an improvement on the existing method of using [4] in full without exploiting the EC, and unlike [11] and [6] the algorithm is valid on curtains.

Proof. The first step, using a single equational constraint, has the same complexity as [6]. Further to this, if the equational constraint has curtains, we need to re-project the non-equational constraints, thus performing a CAD of $m-1$ polynomials of maximum total degree d. Combining these two gives the bound in (1).

The following verifies that the modified algorithm is better than [4] in terms of complexity. It has worse complexity than [11] but is valid for equational constraints with curtains.

The complexity of [4] is $2^{2^n-1}(m+1)^{2^n-2}md^{2^n-1}$ so after removing a factor of $2^{2^n-1}md^{2^n-1}$ the claim is that

$$(3m+1)^{2^{n-1}-1} + (m-1)m^{2^{n-1}-2} < (m+1)^{2^n-2}.$$

But $m \geq 2$ and $n \geq 2$ (otherwise there is nothing to do) so

$$\begin{aligned}
(3m+1)^{2^{n-1}-1} + (m-1)m^{2^{n-1}-2} &< 3^{2^{n-1}-1}(m+1)^{2^{n-1}-1} + (m+1)^{2^{n-1}-1} \\
&= (3^{2^{n-1}-1}+1)(m+1)^{2^{n-1}-1} \\
&= (m+1)^{2^n-2}\frac{3^{2^{n-1}-1}+1}{(m+1)^{2^n-2^{n-1}-1}} \\
&\leq (m+1)^{2^n-2}\frac{3^{2^{n-1}-1}+1}{3^{2^n-2^{n-1}-1}} \\
&= (m+1)^{2^n-2}(3^{-2^n} + 3^{-2^n+2^{n-1}+1}) \\
&< (m+1)^{2^n-2}
\end{aligned}$$

since $3^{-2^n} + 3^{-2^n+2^{n-1}+1} \leq 3^{-4} + 3^{-1} = \frac{28}{81} < 1$.

5 Conclusion and Further research

Algorithm 3 is the first partial CAD algorithm which is a hybrid between sign-invariant and lex-least invariant CAD algorithms. It allows us to exploit equational constraints unconditionally. The novelty lies in performing a second decomposition of the curtain sections of the equational constraints: the worst case for the complexity analysis is when the entire equational constraint is a curtain. More analysis needs to done, as better complexity should be achievable, and we hope to do this the near future. We are currently looking into extending this approach so as to produce as output a partial CAD that has a hybrid of order invariance and lex-least invariance. Note that lex-least invariance over a given region does not imply order invariance, but an order invariant CAD is also lex-least invariant.

References

1. C.W. Brown, M. El Kahoui, D. Novotni, and A. Weber. Algorithmic methods for investigating equilibria in epidemic modeling. *J. Symbolic Comp.*, 41:1157–1173, 2006.
2. G.E. Collins. Quantifier Elimination for Real Closed Fields by Cylindrical Algebraic Decomposition. In *Proceedings 2nd. GI Conference Automata Theory & Formal Languages*, pages 134–183, 1975.
3. Matthew England, Russell Bradford, and James Davenport. Cylindrical algebraic decomposition with equational constraints. *Journal of Symbolic Computation*, 07 2019. doi:10.1016/j.jsc.2019.07.019.
4. D. Lazard. An Improved Projection Operator for Cylindrical Algebraic Decomposition. In C.L. Bajaj, editor, *Proceedings Algebraic Geometry and its Applications: Collections of Papers from Shreeram S. Abhyankar's 60th Birthday Conference*, pages 467–476, 1994.
5. S. McCallum. *An Improved Projection Operation for Cylindrical Algebraic Decomposition*. PhD thesis, University of Wisconsin-Madison Computer Science, 1984.
6. S. McCallum. On Projection in CAD-Based Quantifier Elimination with Equational Constraints. In S. Dooley, editor, *Proceedings ISSAC '99*, pages 145–149, 1999.
7. S. McCallum. On Propagation of Equational Constraints in CAD-Based Quantifier Elimination. In B. Mourrain, editor, *Proceedings ISSAC 2001*, pages 223–230, 2001.
8. S. McCallum. Error in [7]. *E-mail 2019 January 5th*, 2019.
9. S. McCallum, A. Parusiński, and L. Paunescu. Validity proof of Lazard's method for CAD construction. *J. Symbolic Comp.*, 92:52–69, 2019.
10. C.B. Mulligan, J.H. Davenport, and M. England. TheoryGuru: A Mathematica Package to Apply Quantifier Elimination Technology to Economics. In J.H. Davenport, M. Kauers, G. Labahn, and J. Urban, editors, *Proceedings Mathematical Software — ICMS 2018*, pages 369–378, 2018.
11. Akshar Sajive Nair, James Davenport, and Gregory Sankaran. On benefits of equality constraints in lex-least invariant cad (extended abstract. In *Proceedings SC2 2019*, 9 2019.
12. Y. Wada, T. Matsuzaki, A. Terui, and N.H. Arai. An Automated Deduction and Its Implementation for Solving Problem of Sequence at University Entrance Examination. In *Proceedings ICMS 2016*, pages 82–92, 2016.