Equational Constraints, the Lazard Projection and the Curtain Problem

Akshar S. Nair[1], James H. Davenport[1,2] and Gregory Sankaran[2]

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

Abstract

Cylindrical Algebraic Decomposition was introduced by Collins in 1975, to help understand and analyse the real algebraic geometry of a system of polynomials. Nine years later, McCallum's thesis introduced a cheaper algorithm, but it could have problems when polynomials, input or generated, vanished identically (nullified) over a set. Several people have built on McCallum's work to improve the algorithm further when there are equational constraints. Lazard's approach, which was justified in 2019, avoids the nullification problem. The first author investigated translating these equational constraint methods to the Lazard projection. Nullification reappears along subsets called curtains. In this paper we summarise the work done by the first author under the supervision of the other authors for his PhD on propagating the work by Lazard and Brown-McCallum to exploit equational constraints with the help of curtains.

Keywords

Lazard Projection, Cylindrical Algebraic Decomposition, Equational Constraint, Nullification, Curtains

1. Introduction

Cylindrical Algebraic Decomposition (CAD) is a tool for studying real semi-algebraic sets algorithmically. It was introduced by Collins [1] in the context of quantifier elimination, so the precise choice of variables, and their order, matter, and "generic changes of coordinate" are not legitimate. We fix coordinates $x_1, \ldots, x_n$ and regard $\mathbb{R}^k$ as always having coordinates $x_1, \ldots, x_k$. A CAD is a decomposition of a semi-algebraic set $X \subseteq \mathbb{R}^n$ (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 projections of any two cells onto the first k coordinates are either the same or disjoint. We use "QFF" as an abbreviation for "Quantifier-Free Formula", i.e. a Boolean combination of polynomial equalities and inequalities.

Various real-world problems can be reduced to a system of polynomial equalities and inequalities. Often these problems are in the form of a quantifier elimination problem. An example of this is the piano mover's problem [2], [3, §2.8]. There are several specialised algorithms in this area, but CAD is considered one of the most effective methods for quantifier elimination.

CAD algorithms based on [1] consist of three main phases. The first is repeated application of the projection operator, which takes as input a set of polynomials in m variables and outputs a set of polynomials in $m-1$ variables. This operator is used recursively (eliminating variables with respect to our pre-defined order) until the set of polynomials is in the first variable

6^{th} *International Workshop on Satisfiability Checking and Symbolic Computation, 19–20 Aug 2021, College Station U.S.*

© 2021 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings (CEUR-WS.org)

CEUR-WS.org/Vol-3273/paper5.pdf

only. The second phase consists of decomposing $\mathbb{R}^1$ into cells determined by the projected polynomials. The third phase lifts cells recursively from $\mathbb{R}^m$ to $\mathbb{R}^{m+1}$ whilst preserving certain properties. Table 1 describes the various projection operators investigated in [3]. The reductions in coefficients considered in projections doesn't affect the asymptotics, but is of substantial importance in practice, as the gain multiplies at each projection.

Table 1
Projection Operators in [3]

Projection	McCallum (1985b)	Lazard (1994/2016)	Brown-McCallum (2020)
Leading Coefficients	Yes	Yes	Yes
Trailing Coefficients	Yes	Yes	Sometimes
Middle Coefficients	Yes	No	No
Discriminant	Yes	Yes	Yes
Resultant	Yes	Yes	Yes

Collins' original algorithm has bad complexity. McCallum reduced the complexity by switching from sign-invariant CADs to order-invariant CADs. McCallum's improved projection operator is defined as follows.

Definition 1. *Let A be a finite squarefree basis in $\mathbb{R}[x_1, \ldots, x_n]$ with $r \geq 2$. Then McCallum's projection operator $\mathrm{P}(A)$ consists of the following polynomials:*

- *The set of coefficients of all elements of A.*
- *The set of discriminants of all elements of A.*
- *The set of all cross resultants of the elements of A (avoiding the trivial resultants $\mathrm{res}(f, f)$).*

McCallum's algorithm, however, introduced a new problem. Since McCallum's method is based on the calculating the order of a polynomial, the moment a polynomial (input constraint or projected) nullifies, the algorithm will detect failure.

1.1. McCallum with equational constraints

Before we look at the improvements, let us first recall the definition of a variety and an equational constraint.

Definition 2. *A (real)* variety *is a subset of $\mathbb{R}^k$ which is the zero set of one or more polynomials, i.e. $\{(x_1, \ldots x_k) \in \mathbb{R}^k : f_1(x_1, \ldots x_k) = f_2(x_1, \ldots x_k) = \cdots f_l(x_1, \ldots x_k) = 0\}$. We write V_f for $\{(x_1, \ldots x_k) \in \mathbb{R}^k : f(x_1, \ldots x_k) = 0\}$*

Definition 3. *[4] 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 equational constraint is visible in the formula, i.e. the formula Φ is $(f = 0) \wedge \Phi'$, we say the constraint is* syntactically explicit*.*

Example 1 (Equational constraints)**.**

$(f^2 \leq 0) \wedge \Phi'$ *is an explicit equational constraint, but not syntactically explicit. Furthermore the* f^2 *may not be an explicit square.*

$(f \geq 0) \wedge (f \leq 0) \wedge \Phi'$ *does not contain an explicit equational constraint.*

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 exploitable: see also [4].

The main idea behind exploiting syntactically explicit equational constraints is that if the input formula is of the form $(f = 0) \wedge \Phi'$, then to find solutions of the formula it is sufficient to decompose the hypersurface described by $f = 0$ rather than the whole of $\mathbb{R}^n$. In practice f may factor, and in what follows E is the set of factors of f.

Definition 4 (Single Equational Constraint)**.** *[5] Let* A *be a set of pairwise relatively prime polynomials in* $\mathbb{R}[x_1, \ldots, x_n]$ *with* $n \geq 2$ *and let* $E \subset A$*. McCallum's restricted projection operator* $\mathrm{P}_E(A)$ *is defined as follows:*

$$\mathrm{P}_E(A) = \mathrm{P}(E) \cup \{\mathrm{res}_{x_n}(f, g) \mid f \in E,\ g \in A \setminus E\}. \tag{1}$$

[5] showed that, if we have an equational constraint $f = 0$ involving x_n, it suffices to use $\mathrm{P}_E(A)$ to eliminate x_n, then proceed as usual to eliminate down to x_1, then lift back to an order-invariant decomposition of $\mathbb{R}^{n-1}$, which can then be lifted to a decomposition of $\mathbb{R}^n$ such that the input polynomials are sign-invariant on V_f. This improved on the complexity of [9] since it uses significantly fewer polynomials, which also results in the creation of fewer cells. However, this algorithm fails if the EC nullifies anywhere, or if nullification causes a problem with the lift to $\mathbb{R}^{n-1}$. The second drawback to this method is that this projection operator cannot be used recursively. This is because this method lifts an order-invariant decomposition to a sign-invariant decomposition.

In 2001 [6], McCallum suggested a further extension of his method from [5], to support multiple equational constraints, i.e. $(f_1 = 0) \wedge (f_2 = 0) \wedge \cdots \wedge \Phi'$. The new projection operator supporting this extension is defined as follows. We designate one of the equational constraints, say $f = 0$, as the appropriate constraint at this level, and in what follows E is the set of factors of f.

Definition 5 (Multiple Equational Constraints)**.** *[6] Let* A *be a set of pairwise relatively prime polynomials in* $\mathbb{R}[x_1, \ldots, x_n]$ *with* $n \geq 2$ *and let* $E \subset A$*. McCallum's semi-restricted projection operator* $\mathrm{P}^*_E(A)$ *is defined as follows:*

$$\mathrm{P}^*_E(A) = \mathrm{P}(A) \setminus \mathrm{res}(A \setminus E, A \setminus E). \tag{2}$$

Note that $\mathrm{P}^*_E(A)$ includes the discriminants of $A \setminus E$, which $\mathrm{P}_E(A)$ does not.

This operator can be used recursively as it lifts order invariance to order invariance. However, because it uses order invariance, it can still fail because of nullification.

2. Lazard Projection and Lifting

This method, adjusting the lifted polynomials when nullification occurred, was introduced in [8].

Definition 6. *Let $v, w \in \mathbb{Z}^n$. We say that $v = (v_1, \ldots, v_n) \geq_{lex} (w_1, \ldots, w_n) = w$ if and only if either $v = w$ or there is 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 7. *[7, 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 $\geq_{lex}$) 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 $c \neq 0$.

Definition 8. *[7] 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 semi-valuation $\nu'_\beta(f) = (\nu_1, \ldots, \nu_{n-1})$ 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, $(\nu_1, \ldots, \nu_{n-1})$

Definition 9. *[7, Definition 2.10] Let S be a connected subset of $\mathbb{R}^{n-1}$ and $f \in \mathbb{R}[x_1, \ldots, x_n]$. We say that f is Lazard delineable on S if all the following hold.*

i) *The lex-least semi-valuation of f at β is the same for each point $\beta \in S$.*
ii) *There exist finitely many (possibly zero) continuous functions $\theta_i \colon S \to \mathbb{R}$, such that for all $\beta \in S$, $\theta_1(\beta) < \ldots < \theta_k(\beta)$, and the set of real roots of f_β is $\{\theta_1(\beta), \ldots, \theta_k(\beta)\}$.*
iii) *If $k = 0$, then for all $\beta \in S$ the set of real roots of f_β is empty. 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.*

With the definition of delineability, we are able to define Lazard sections and sectors.

Definition 10. *[7, Definition 2.10] Let f be Lazard delineable on $S \subseteq \mathbb{R}^{n-1}$.*

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

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

To combine Lazard's method with [5], the following definition is introduced in [3].

Definition 11. *Let A be a finite set of irreducible polynomials in $\mathbb{R}[x_1, \ldots, x_n]$ with $n \geq 2$ and let E be a subset of A. The modified Lazard projection operator $\mathrm{PL}_E(A)$ is the subset of $\mathbb{R}[x_1, \ldots, x_{n-1}]$ consisting of the following polynomials:*

- *All leading coefficients of the elements of E.*
- *All trailing coefficients of the elements of E.*
- *All discriminants of the elements of E.*
- *All resultants of pairs of distinct elements of E.*
- *All resultants $\mathrm{res}_{x_n}(f, g)$ where $f \in E$ and $g \in A \setminus E$.*

We can also define it as follows:

$$\mathrm{PL}_E(A) = \mathrm{PL}(E) \cup \{\mathrm{res}_{x_n}(f, g) \mid f \in E,\, g \in A \setminus E\}.$$

3. Curtains

Lazard [8, justified in [7]] removed the nullification limitation by decomposing according to the lex-least valuation, rather than the order valuation used in [9]. This in effect allows us to identify the nullifying factors and disregard them in the lifting phase. The first author's thesis [3] (see also [10]) shows how to adapt these methods so as to exploit equational constraints: however, nullification then reappears as a problem, in a different guise. To understand this we shift attention to the geometric loci where nullification occurs, which we call curtains.

Definition 12. *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 $y \in \mathbb{R}$.*

In other words, C is a curtain if it is a union of fibres of $\mathbb{R}^n \to \mathbb{R}^{n-1}$, i.e. is $\hat{C} \times \mathbb{R}$ for a variety $\hat{C} \subset \mathbb{R}^{n-1}$.

Definition 13. *Suppose $f \in \mathbb{R}[x_1, \ldots, x_n]$ and $S \subseteq \mathbb{R}^{n-1}$. We say that V_f (or f) has a curtain at S if for all $(\alpha_1, \ldots, \alpha_{n-1}) \in S$ and for all $y \in \mathbb{R}$ we have $f(\alpha_1, \ldots, \alpha_{n-1}, y) = 0$. We call S the* base *of the curtain.*

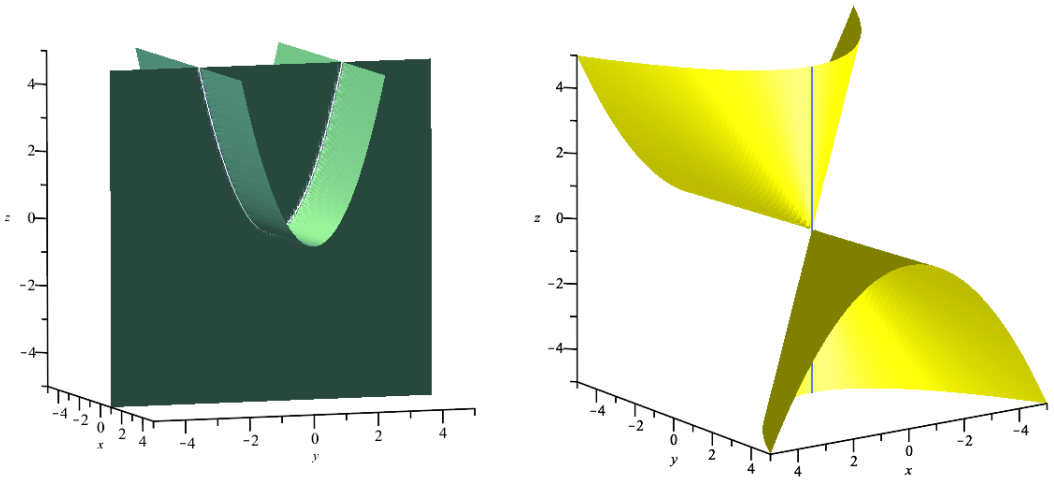
For a hypersurface V_f there are two types of curtain.

Definition 14. *Let $f \in \mathbb{R}[x_1, \ldots, x_n]$ and suppose that C is a curtain contained in V_f. Then C is called an* explicit curtain *if f factorises as $f = gh$, where $g \in \mathbb{R}[x_1, \ldots, x_{n-1}]$ and $C = \{x \in \mathbb{R}^n \mid g(x) = 0\} \subset \mathbb{R}^n$. In this case the base of the curtain is given by $g = 0$ in $\mathbb{R}^{n-1}$. Otherwise C is called an* implicit curtain.

Note that, if C is an explicit curtain, it is possible that h itself has a curtain, explicit or implicit, which may include some or all of C. Explicit curtains arise when f has non-trivial content, an obstacle already noted in [11, 12], but implicit curtains are also a challenge, and are not so easily detected.

Example 2 (Curtains). *See Figure 1.*

- *Explicit Curtain:* $f(x, y, z) = xy^2 - y^2 - xz + z = (x - 1)(y^2 - z)$, *curtain at* $(1, y, z)$. *The curtain can be seen in Figure 1a as the sheet given by* $x = 1$.
- *Implicit Curtain:* $f(x, y, z) = x^2 + yz$, *curtain at* $(0, 0, z)$. *This can be seen in Figure 1b, where the blue line represents the curtain:* $x = y = 0$.



(a) Surface with an Explicit Curtain

(b) Surface with an Implicit Curtain

Figure 1: Different types of curtains

To deal with curtains in the lifting phase with equational constraints, we first need to be able to identify where curtains occur. Our process for doing so relies on the following lemma, which establishes a relationship between curtains and the lex-least valuation.

Lemma 1. *Let* $f \in \mathbb{R}[x_1, \ldots, x_n]$ *and* $\alpha \in \mathbb{R}^{n-1}$. *Then* $\nu'_\alpha(f) \neq 0$ *if and only if there is a curtain above* α.

Definition 15. *Let* $f \in \mathbb{R}[x_1, \ldots, x_n]$ *and* $\alpha \in \mathbb{R}^{n-1}$. *We say that* f *has a point curtain at* α *if*

- $f(\alpha, y) = 0$ *for all* $y \in \mathbb{R}$, *and*
- *there exists a Euclidean open neighbourhood* $U \subset \mathbb{R}^{n-1}$ *of* α *such there exists no* $\beta \in U \setminus \{\alpha\}$ *such that* $f(\beta, y) = 0$ *for all* $y \in \mathbb{R}$.

Nair [3, Ch. 6–7] shows that point curtains do not disrupt the lifting phase in Lazard's algorithm. Algorithm 2 describes how point curtains are detected during lifting.

Algorithm 2 Detecting and Classifying Curtains

$(B, B') \leftarrow PC(f, I, S, n)$

Input = Set of indices I, set of sample points S with respect to the indices I which correspond to the CAD cells in $\mathbb{R}^{n-1}$, equational constraint $f \in \mathbb{R}[x_1, \ldots, x_n]$.

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 in curtains (but not point curtains).

1: $B \leftarrow$ Empty list.
2: $B' \leftarrow$ Empty list.
3: **for** $\alpha \in S$ **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').

4. Complexity analysis

Simple time-step complexity analysis is not sufficient to compare the various projection operators that have been proposed. We use the (m, d)-property introduced by McCallum [9, Section 6.1] and improved by England *et al.* [4]. We also modify these definitions and methods to account for the effect of equational constraints.

Definition 16. *[9, Section 6.1] A set of polynomials has the* (m, d)-property *if it can be partitioned into* m *sets, such that the maximum (in each variable) degree of the product of each set is less than or equal to* d*: that is* $\max_i \deg_{x_i} \prod_{f \in S} f \leq d$.

When projecting with equational constraints we need to use an enhanced version of the (m, d)-property, so that any statement made about projection operators using equational constraints can be used recursively.

Definition 17. *[3, Chapter 8] Let* A *be a set of polynomial factors of a family of polynomial constraints. We say that* A *has the* $(m, d)_k$-property *if* A *can be written as the union of* m *sets each of combined maximum degree* $\leq d$ *and each of the first* k *sets consist of the factors of a single equational constraint.*

The outcome of the complexity analysis carried out in [3] is summarised in Table 2. If the set of inputs has the (m, d)-property then the set of projected polynomials after projection has the $(M, 2d^2)$-property, with M as in Table 2.

This shows that, if we have only one equational constraint, the "Single EC" methods are better. However, if we have multiple equational constraints, the "Multiple EC" methods are better: for projections with two (or more) equational constraints, the number of polynomials in $n - 2$ variables is $O(m^4)$ for the original methods, $O(m^2)$ for the "Single EC" methods and $O(m)$ for the "multiple EC" methods.

Table 2
Growth of polynomials in CAD

Base Theory by	Constraints	Original	Single EC	Multiple EC
McCallum	[5, 6]	$\left\lfloor \frac{(m+1)^2}{2} \right\rfloor$	$\left\lfloor \frac{5m+4}{4} \right\rfloor$	$\left\lfloor \frac{11m}{4} \right\rfloor$
Lazard	[3]	$\left\lfloor \frac{(m+1)^2}{2} \right\rfloor$	$\left\lfloor \frac{5m+3}{4} \right\rfloor$	$\left\lfloor \frac{9m-1}{4} \right\rfloor$
Brown-McCallum	From [3]	$\frac{m(m+1)}{2}$	$\left\lfloor \frac{5m+2}{4} \right\rfloor$	$\left\lfloor \frac{9m-2}{4} \right\rfloor$

5. Conclusion and Some Future Work

There have been various improvements to the implementations of CAD algorithms to further decrease the space and time complexity. In this abstract we have given a brief account of the work done by Nair et al. in [10, 3] on what caused the problems for propagating equational constraints and how to circumvent them. See Table 2 for a summary of the effect on the number of polynomials.

There are several avenues one can take to extend this work. Some implementation of the Lazard-based algorithms has been done in [13] (and there are some useful points in [14]). However, an implementation of the modified Brown-McCallum algorithm [15] for equational constraints is yet to be done. Such an implementation would work with curtains.

Exploring how curtains can be detected prior to the projection phases is still an open question. Such an algorithm would be extremely helpful, as it could be used to eliminate bad candidates for designated equational constraints.

Finally there is the important and completely open question of whether the lex-least valuation is the best choice for computing valuation-invariant CADs. Are there other valuations that could deal with curtains in the input constraints more appropriately? If so, can they be used to take advantage of equational constraints?

References

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

[2] D. Wilson, R. Bradford, J. Davenport, M. England, A "Piano Movers" Problem Reformulated, Technical Report CSBU-2013-03 Department of Computer Science University of Bath, 2013.

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

[4] M. England, R. Bradford, J. Davenport, Cylindrical Algebraic Decomposition with Equational Constraints, in: J. Davenport, M. England, A. Griggio, T. Sturm, C. Tinelli (Eds.),

Symbolic Computation and Satisfiability Checking: special issue of Journal of Symbolic Computation, volume 100, 2020, pp. 38–71.

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

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

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

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

[9] S. McCallum, An Improved Projection Operation for Cylindrical Algebraic Decomposition, Technical Report 578, Computer Science University Wisconsin at Madison, 1985. URL: https://minds.wisconsin.edu/bitstream/handle/1793/58594/TR578.pdf?sequence=1.

[10] S. McCallum, A. Nair, J. Davenport, G. Sankaran, The CAD Conundrum: Lex-Least vs Order, in: Proceedings 2020 22nd International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, 2021, pp. 32–35.

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

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

[13] Z. Tonks, Poly-algorithmic Techniques in Real Quantifier Elimination, Ph.D. thesis, University of Bath, 2021. URL: https://researchportal.bath.ac.uk/en/studentTheses/poly-algorithmic-techniques-in-real-quantifier-elimination.

[14] G. Kremer, J. Brandt, Implementing arithmetic over algebraic numbers: A tutorial for Lazard's lifting scheme in CAD, 23rd International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2021) (2021) 4–10.

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