Complexity of Cylindrical Algebraic Decompositions via Regular Chains

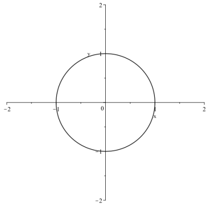
Corin Lee

Department of Mathematical Sciences
University of Bath

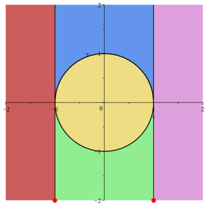
What is a CAD?

A *cylindrical algebraic decomposition* (CAD) is a method in real algebraic geometry to decompose $\mathbb{R}^n$ into a finite number of disjoint cells, each homeomorphic to some $\mathbb{R}^k$, over which a given set of polynomials have constant sign. Given such a decomposition it is easy to give a solution of a system of inequalities and equations defined by the polynomials.

It has applications in areas such as robotics and air traffic control problems, and was first discovered by George E. Collins in 1973, as part of his work on an effective method for quantifier elimination in real closed fields (a concept involving replacing a formula with an equivalent one without quantifiers).



$\longrightarrow$



Definitions

A cylindrical algebraic decomposition is in fact a decomposition that is both cylindrical and algebraic:

Definition

- **Decomposition:** Let $X \subseteq \mathbb{R}^n$. A *decomposition* of X is a finite collection of disjoint cells whose union is X.
- **Cylindrical:** A decomposition is *cylindrical* if for all $1 \leq j < n$, the projections on the first j variables of any two cells are either equal or disjoint.
- **Algebraic:** A decomposition is *algebraic* if each of its cells is a *semi-algebraic set*, i.e. a set that can be constructed by finitely many unions, intersections and complementations on sets of the form $\{x \in \mathbb{R}^n \mid f(x) \geq 0\}$, where $f \in \mathbb{R}[x_1, \ldots, x_n]$.

Definition

Let $X \subseteq \mathbb{R}^n$ and $\mathbb{R}[x_1, \ldots, x_n] \ni \mathcal{F} = \{f_i \in k[x_1, \ldots, x_n], 1 \leq i \leq r\}$, we say X is $\mathcal{F}$*-invariant* if each $f_i(x)$ has constant sign for every $x \in X$, that is, $\forall x \in X : f_i(x) \ \diamond \ 0$ for $\diamond = \{>, =, <\}$.

The CAD Algorithm

Collins' original CAD method is based on a projection and lifting scheme (PL-CAD):

- **Projection:** Repeatedly apply a projection operator *Proj*:

$$\mathcal{F} = \mathcal{F}_n(x_1, \ldots, x_n) \xrightarrow{Proj} \mathcal{F}_{n-1}(x_1, \ldots, x_{n-1}) \xrightarrow{Proj} \ldots \xrightarrow{Proj} \mathcal{F}_1(x_1)$$

  The zero sets of the polynomials produced by each step are the projections of "significant points" of the previous set of polynomials.

- **Base phase:** Real root isolation on $\mathcal{F}_1(x_1)$, roots and open intervals between form an $\mathcal{F}_1$-invariant CAD of $\mathbb{R}^1$.
- **Lifting phase:** For each cell C of the $\mathcal{F}_{k-1}$-invariant CAD in $\mathbb{R}^{k-1}$, construct a sample point s and isolate the real roots of the polynomials of $\mathcal{F}_k$ at s. The sectors and sections of these polynomials form a stack, and these stacks make the cells of the $\mathcal{F}_k$-invariant CAD of $\mathbb{R}^k$ above C.

It is enough to determine the signs of $\mathcal{F}$ in these sample points as each cell is $\mathcal{F}$-invariant by construction.

The Projection Operator

So what is this projection operator, what are these “significant points” and how are they calculated?

In short, the set of projection polynomials of $\mathcal{F}$ involve points such as vertical tangents of each polynomials and crossings between them. Fully describing the projection operator would involve defining another dozen things (to the point where many papers omit it!), but we will focus on some key components.

In a simple sense, for $\mathbf{x} = x_1 < \cdots < x_n$ and $\mathcal{F} = \{f_i \in \mathbb{R}\left[\mathbf{x}\right], 1 \leq i \leq r\} \subseteq \mathbb{R}[\mathbf{x}]$, takes each polynomial $f_i \in \mathcal{F}$ and treats it as a univariate polynomial in its greatest variable, i.e. $f_i \in \mathbb{R}\left[x_1, \ldots, x_{n-1}\right][x_n]$. It then looks at the coefficients of each polynomial (in $\mathbb{R}[x_1, \ldots, x_{n-1}]$), as well as their *resultants* and *discriminants*.

An Example in Three Dimensions

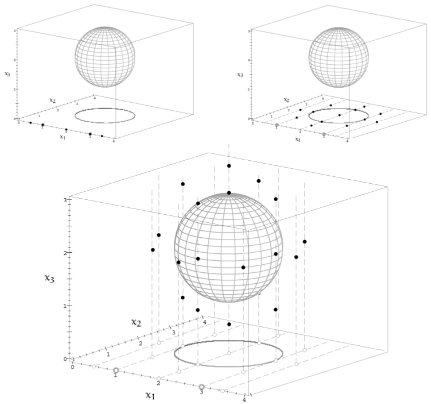
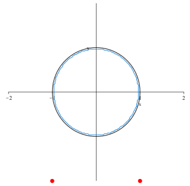


Figure: Sample points for the unit sphere centred at $(2, 2, 2)$,
$f = (x_1 - 2)^2 + (x_2 - 2)^2 + (x_3 - 2)^3 - 1$.

A very simple example

We will now look step by step at constructing a CAD of $\mathbb{R}^2$ for the previously-seen example of the unit circle, $f = x^2 + y^2 - 1$. You can tell I'm proud of the pictures. Recall this is treated as a polynomial in y.



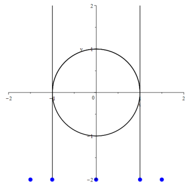
The Projection Phase

- Coefficients: $\{1, x^2 - 1\}$.
- Resultants: None!
- Discriminants: $disc(f) = \begin{vmatrix} 1 & 0 & x^2 - 1 \\ 2 & 0 & 0 \\ 0 & 2 & 0 \end{vmatrix} = 4(x^2 - 1)$.

The only real zeroes of these polynomials are $x = \pm 1$, which correspond to the vertical tangents.

A very simple example

We will now look step by step at constructing a CAD of $\mathbb{R}^2$ for the previously-seen example of the unit circle, $f = x^2 + y^2 - 1$. This is treated as a polynomial in y.



The Base Phase

We now have the stack over $\mathbb{R}^1$: $(-\infty, -1), -1, (-1, 1), 1, (1, \infty)$ and thus a CAD with those cells.

We can then take some "nice" sample points. In the picture we can see that the x-axis is now decomposed into five cells with with sample points $-1.5, -1, 0, 1, 1.5$.

A very simple example

We will now look step by step at constructing a CAD of $\mathbb{R}^2$ for the previously-seen example of the unit circle, $f = x^2 + y^2 - 1$.

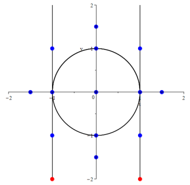


Figure: We pick sample points for each cell of $\mathbb{R}^2$.

The Lifting Phase

Substitute these sample points into f:

$x = -1.5:$	$f = y^2 + 1.25$	$+$ always	(1 region)
$x = -1:$	$f = y^2$	$+$ when $y > 0, \quad y < 0$	
		0 when $y = 0,$	(3 regions)
$x = 0:$	$f = y^2 - 1$	$+$ when $y > 1, \quad y < -1,$	
		0 when $y = 1, \quad y = -1,$	
		$-$ when $-1 < y < 1,$	(5 regions)
$x = 1:$	$f = y^2$	$+$ when $y > 0, \quad y < 0$	
		0 when $y = 0,$	(3 regions)
$x = 1.5:$	$f = y^2 + 1.25$	$+$ always	(1 region)

That means there are 13 regions in total. If we choose some nice sample points, we're done!

A very simple example

We can see that even for such a simple example, we have decomposed $\mathbb{R}^2$ into 13 cells!

Using *Maple*, we can see descriptions of all cells:

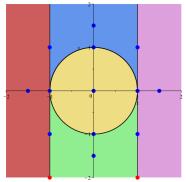


Figure: The f-invariant CAD of $\mathbb{R}^2$.

$$
cad := \begin{cases}
1 & x < -1 \\
\begin{cases} 1 & y < 0 \\ 1 & y = 0 \\ 1 & 0 < y \end{cases} & x = -1 \\
\begin{cases} 1 & y < -\sqrt{-x^2+1} \\ 1 & y = -\sqrt{-x^2+1} \\ 1 & -\sqrt{-x^2+1} < y \wedge y < \sqrt{-x^2+1} \\ 1 & y = \sqrt{-x^2+1} \\ 1 & \sqrt{-x^2+1} < y \end{cases} & -1 < x \wedge x < 1 \\
\begin{cases} 1 & y < 0 \\ 1 & y = 0 \\ 1 & 0 < y \end{cases} & x = 1 \\
1 & 1 < x
\end{cases}
$$

A Different CAD Algorithm

There has been much work by Chen and Moreno Maza on an incremental algorithm for computing CADs using triangular systems and regular chains (RC-CAD), by creating a complex cylindrical tree and refining it into a real cylindrical tree.

A lot has been learnt about PL-CAD over the last forty years, with many enhancements made along the way. However, there has been a lot of work implementing RC-CAD into *Maple* via the `RegularChains` package, little analysis has been done. Therefore we are interested in trying to understand how the algorithm works in sufficient detail to give estimates and boundaries for its complexity, and in learning when and where the algorithm is more, or less, efficient to give some kind of heuristic for when it is better to use RC-CAD or PL-CAD.

References

Collins, George (1975)
Quantifier elimination for real closed fields by cylindrical algebraic decomposition
Lecture Notes in Computer Science.

Arnon, Dennis and Collins, George and McCallum, Scott (1984)
Cylindrical Algebraic Decomposition I: The Basic Algorithm
SIAM J. Comput. 13(11), 865 – 877.

Jirstrand, Mats (1995)
Cylindrical Algebraic Decomposition - an Introduction

Chen, Changbo and Moreno Maza, Marc (2009)
Computing cylindrical algebraic decomposition via triangular decomposition
Proceedings of the 2009 international symposium on Symbolic and algebraic computation, 95 – 102.

Chen, Changbo and Moreno Maza, Marc (2012)
An Incremental Algorithm for Computing Cylindrical Algebraic Decompositions