Cylindrical Algebraic Decomposition in *Macaulay2*

Corin Lee

Department of Mathematical Sciences
University of Bath

International Congress on Mathematical Software
July 23, 2024

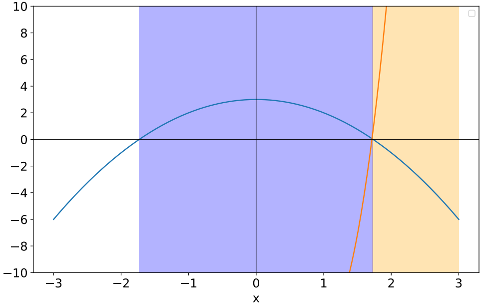
Overview

- **Summary:** An introduction to Cylindrical Algebraic Decomposition and the upcoming `CADecomposition` package for *Macaulay2*.
- **Joint work with:** Tereso del Río and Hamid Rakhooy.

Motivation

Does there exist an $x \in \mathbb{R}$ such that

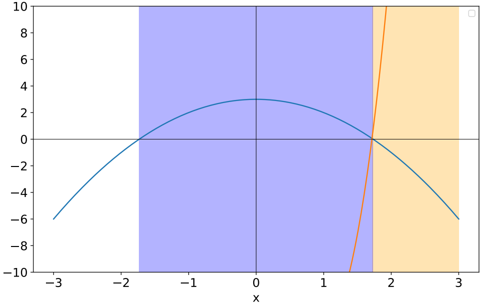
$$g(x) = 3 - x^2 > 0 \quad \text{and} \quad h(x) = (7x - 12)(x^2 + x + 1) > 0 \,? \tag{1}$$



Motivation

Does there exist an $x \in \mathbb{R}$ such that

$$g(x) = 3 - x^2 > 0 \quad \text{and} \quad h(x) = (7x - 12)(x^2 + x + 1) > 0\,? \tag{1}$$

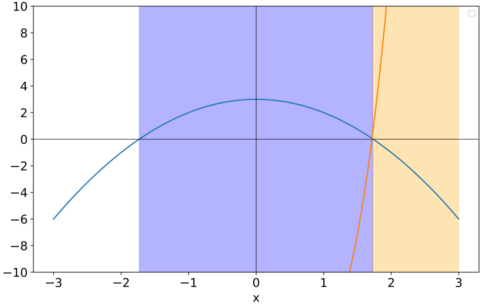


Numerical approach: Sample 1001 equispaced points between -50 and 50.

Motivation

Does there exist an $x \in \mathbb{R}$ such that

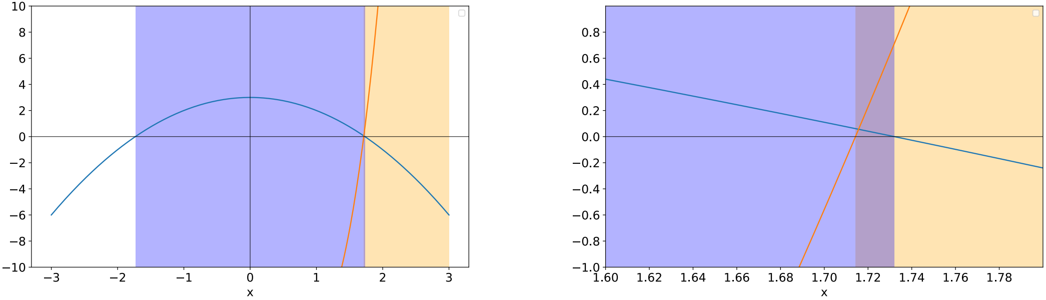
$$g(x) = 3 - x^2 > 0 \quad \text{and} \quad h(x) = (7x - 12)(x^2 + x + 1) > 0 ? \tag{1}$$



Numerical approach: Sample 1001 equispaced points between -50 and 50. No point found.

Motivation

However, such a point does exist:



Drawbacks to doing this numerically:

- How many samples needed to guarantee it is found?
- If it does not exist, how would you know?

Motivation

Symbolic approach: Real *quantifier elimination* (QE):

Quantified:	$\exists x\ \left(3 - x^2 > 0\right) \wedge \left((7x - 12)(x^2 + x + 1) > 0\right)$
Quantifier-free:	$12/7 < x < \sqrt{3}$

Motivation

Symbolic approach: Real *quantifier elimination* (QE):

Quantified:	$\exists x \ \left(3 - x^2 > 0\right) \wedge \left((7x - 12)(x^2 + x + 1) > 0\right)$
Quantifier-free:	$12/7 < x < \sqrt{3}$

Can solve real QE problems (i.e. find the equivalent quantifier-free formula) by constructing a *sign-invariant Cylindrical Algebraic Decomposition* (CAD), reducing the search to a finite number of *cells* in which the signs of the polynomials, i.e. -,0,+, do not change.

Motivation

Symbolic approach: Real *quantifier elimination* (QE):

$$\begin{array}{ll} \textbf{Quantified:} & \exists x\ \left(3 - x^2 > 0\right) \wedge \left((7x-12)(x^2+x+1) > 0\right) \\ \textbf{Quantifier-free:} & 12/7 < x < \sqrt{3} \end{array}$$

Can solve real QE problems (i.e. find the equivalent quantifier-free formula) by constructing a *sign-invariant Cylindrical Algebraic Decomposition* (CAD), reducing the search to a finite number of *cells* in which the signs of the polynomials, i.e. -,0,+, do not change.

$$\begin{aligned} CAD(\{g,h\}) &= \{x < -\sqrt{3}\} \cup \{x = -\sqrt{3}\} \cup \{x > -\sqrt{3} \wedge x < 12/7\} \\ &\cup \{x = 12/7\} \cup \{x > 12/7 \wedge x < \sqrt{3}\} \cup \{x = \sqrt{3}\} \cup \{x > \sqrt{3}\}. \end{aligned}$$

Motivation

Using the package `CADecomposition`, we are able to construct a (weaker form of a) CAD and identify such a sample point:

```
i1 : R=QQ[x];
i2 : findPositiveSolution({3-x^2,(7*x-12)*(x^2+x+1)})
o2 = (true, HashTable{x => 1539/896})
```

This package implements a sign-invariant, *projection and lifting* open CAD, and:

- Is the first implementation of CAD in *Macaulay2*.
- Uses the contemporary Lazard projection [Laz94; MPP19].
- If the first implementation of the `gmods` heuristic for variable ordering [dE22].

Cylindrical Algebraic Decomposition

First presented in [Col75] as an algorithm to tackle real QE problems.

Applications in various fields including robotics [Man+12], economics [MDE18] and biology [RS21].

Input: Set of polynomials $\mathcal{F}_n = \{f_1, \ldots, f_r\}$ in variables $x_1, \ldots, x_n$.

Output (CAD): CAD($\mathcal{F}_n$), a decomposition of $\mathbb{R}^n$ into *cells* which are sign-invariant with respect to $\mathcal{F}_n$, described by polynomial constraints and are cylindrically arranged.

Output (`CADecomposition`): $\mathcal{S}_n$, a tree of sample points in n variables which represent each *open cell* of CAD($\mathcal{F}_n$).

Cylindrical Algebraic Decomposition

- **Decomposition** of $\mathbb{R}^n$ into finitely many *cells* C_i where $\bigcup C_i = \mathbb{R}^n$, $C_i \cap C_j = \emptyset$ if $i \neq j$,

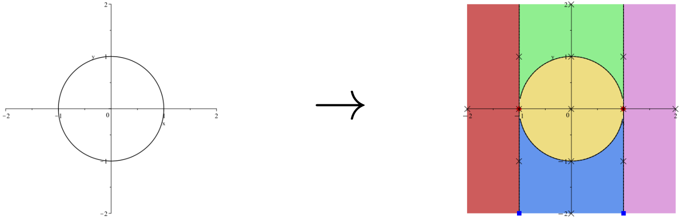


Figure: The graph of $f = x^2 + y^2 - 1$ and an associated sign-invariant CAD of $\mathbb{R}^2$.

Cylindrical Algebraic Decomposition

- **Decomposition** of $\mathbb{R}^n$ into finitely many *cells* C_i where $\bigcup C_i = \mathbb{R}^n$, $C_i \cap C_j = \emptyset$ if $i \neq j$,
  - and for $0 \leq j \leq n$, a *j-cell* in $\mathbb{R}^n$ is a subset of $\mathbb{R}^n$ that is homeomorphic to $\mathbb{R}^j$.

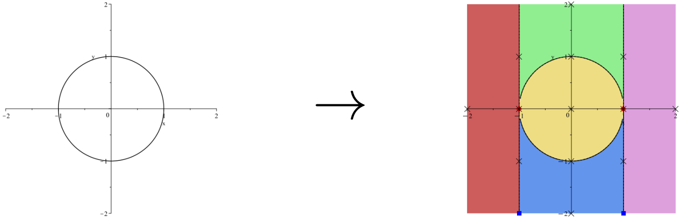


Figure: This CAD is made up of 13 cells: five 2-cells, six 1-cells and two 0-cells. The black crosses represent the sample points for each cell.

Cylindrical Algebraic Decomposition

- **Decomposition** of $\mathbb{R}^n$ into finitely many *cells* C_i where $\bigcup C_i = \mathbb{R}^n$, $C_i \cap C_j = \emptyset$ if $i \neq j$,
  - and for $0 \leq j \leq n$, a j-*cell* in $\mathbb{R}^n$ is a subset of $\mathbb{R}^n$ that is homeomorphic to $\mathbb{R}^j$.
- **Cylindrical:** The *projections* of any two cells into lower-dimensional space are either equal or disjoint (they 'stack up' over lower-dimensional cells).

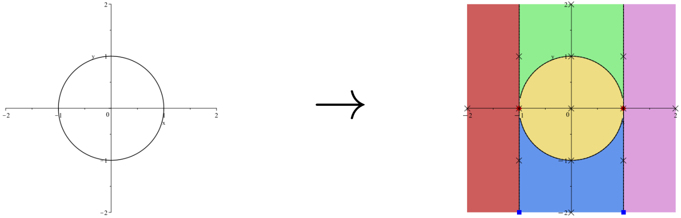


Figure: These cells stack in *cylinders* over the 5 cells of the CAD of $\mathbb{R}^1$ (the two points at the bottom and the regions between them).

Cylindrical Algebraic Decomposition

- **Decomposition** of $\mathbb{R}^n$ into finitely many *cells* C_i where $\bigcup C_i = \mathbb{R}^n$, $C_i \cap C_j = \emptyset$ if $i \neq j$,
  - and for $0 \leq j \leq n$, a j-*cell* in $\mathbb{R}^n$ is a subset of $\mathbb{R}^n$ that is homeomorphic to $\mathbb{R}^j$.
- **Cylindrical:** The *projections* of any two cells into lower-dimensional space are either equal or disjoint (they 'stack up' over lower-dimensional cells).
- **Algebraic:** Each cell can be described by a finite set of polynomial equations and inequalities.

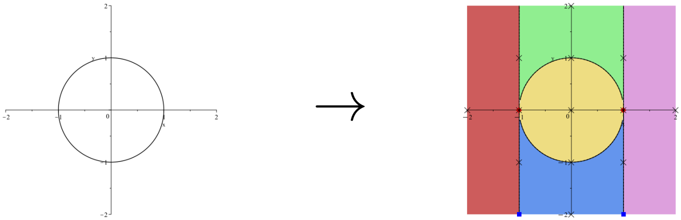


Figure: Each cell is described by constraints on x and y.

Projection and Lifting

Projection Phase: Apply a *projection operator* Proj to $\mathcal{F}_n$ to obtain a new set $\mathcal{F}_{n-1}$ in $n-1$ variables. Repeat this until one obtains $\mathcal{F}_1$.

$$\begin{array}{c} \mathcal{F}_n \\ \big\downarrow \mathrm{Proj} \\ \mathcal{F}_{n-1} \\ \big\downarrow \mathrm{Proj} \\ \vdots \\ \big\downarrow \mathrm{Proj} \\ \mathcal{F}_2 \\ \big\downarrow \mathrm{Proj} \\ \mathcal{F}_1 \end{array}$$

Projection and Lifting

Projection Phase: Apply a *projection operator* Proj to $\mathcal{F}_n$ to obtain a new set $\mathcal{F}_{n-1}$ in $n-1$ variables. Repeat this until one obtains $\mathcal{F}_1$.

Base Phase: Decompose $\mathbb{R}^1$ into the roots of each polynomial in $\mathcal{F}_1$ and the open intervals either side of them to obtain $\mathrm{CAD}(\mathbb{R}^1)$.

$$\begin{array}{l} \mathcal{F}_n \\ \quad\downarrow \text{Proj} \\ \mathcal{F}_{n-1} \\ \quad\downarrow \text{Proj} \\ \quad\vdots \\ \quad\downarrow \text{Proj} \\ \mathcal{F}_2 \\ \quad\downarrow \text{Proj} \\ \mathcal{F}_1 \xrightarrow[\text{split into cells}]{\text{find roots}} \end{array}$$

Projection and Lifting

Projection Phase: Apply a *projection operator* Proj to $\mathcal{F}_n$ to obtain a new set $\mathcal{F}_{n-1}$ in $n-1$ variables. Repeat this until one obtains $\mathcal{F}_1$.

Base Phase: Decompose $\mathbb{R}^1$ into the roots of each polynomial in $\mathcal{F}_1$ and the open intervals either side of them to obtain $\mathrm{CAD}(\mathbb{R}^1)$.

Lifting Phase: For each cell C of $\mathrm{CAD}(\mathbb{R}^1)$, substitute the variable constraints into $\mathcal{F}_2$ to obtain a new set of univariate polynomials. Decompose these to obtain a set of cells that lie above C, and the full collection of these cells for every C forms $\mathrm{CAD}(\mathbb{R}^2)$. Repeat this process until one obtains $\mathrm{CAD}(\mathbb{R}^n)$.

$$
\begin{tikzcd}
\mathcal{F}_n \arrow[d, "\mathrm{Proj}"] & \mathrm{CAD}(\mathcal{F}_n) \\
\mathcal{F}_{n-1} \arrow[d, "\mathrm{Proj}"] & \mathrm{CAD}(\mathcal{F}_{n-1}) \arrow[u, "\mathrm{Lift}"'] \\
\vdots \arrow[d, "\mathrm{Proj}"] & \vdots \arrow[u, "\mathrm{Lift}"'] \\
\mathcal{F}_2 \arrow[d, "\mathrm{Proj}"] & \mathrm{CAD}(\mathcal{F}_2) \arrow[u, "\mathrm{Lift}"'] \\
\mathcal{F}_1 \arrow[r, "\text{find roots}", "\text{split into cells}"'] & \mathrm{CAD}(\mathcal{F}_1) \arrow[u, "\mathrm{Lift}"']
\end{tikzcd}
$$

Projections and Variable Ordering

The CAD has some *variable ordering* $x_1 \prec x_2 \prec \cdots \prec x_{n-1} \prec x_n$ such that $\mathcal{F}_k \subset \mathbb{R}[x_1, \ldots, x_k]$ for all $1 \leq k \leq n$ and x_k is 'projected away' when moving from $\mathcal{F}_k$ to $\mathcal{F}_{k-1}$.

This is done by treating the polynomials in $\mathcal{F}_k$ as univariate in x_k with coefficients in $\mathbb{R}[x_1, \ldots, x_{k-1}]$.

Variable ordering chosen can have significant impact on the number of cells in a CAD and thus the resources and time needed to produce it, thus considerable work has been done trying to find the best way to choose the variable ordering [Hua+15; EF19; CZC20].

One of the simplest and best working strategies is `gmods`, developed by Del Río and England [dE22], which chooses to project first *"the variable with the lowest degree sum [...] in the set of polynomials"* i.e. the variable x such that $\sum_i \deg_x(f_i)$ is smallest. This is the strategy used in `CADecomposition`.

The Lazard Projection operator

The projection operator produces a set of polynomials whose roots represent the 'significant points' of the previous set of polynomials.

This package uses the *Lazard projection*, a projection operator originally proposed in [Laz94] and later validated in [MPP19]:

Definition (Lazard projection)

Let $\mathcal{F}_n$ be a finite set of irreducible pairwise relative prime polynomials in $\mathbb{R}[x_1, \ldots, x_n]$, $n \geq 2$. The *Lazard projection* of $\mathcal{F}_n$, $P_L(\mathcal{F}_n) \subset \mathbb{R}[x_1, x_2, \ldots, x_{n-1}]$, comprises the following:

1. all leading coefficients of each f_i, (behaviour at infinity)
2. all trailing coefficients (i.e. independent of x_n) of each f_i, (behaviour at 0)
3. all discriminants of each f_i, (self-crossings)
4. all resultants of pairs of distinct elements of $\mathcal{F}_n$. (common roots)

In our code, we ensure that the input and output polynomials are irreducible and pairwise coprime by replacing the polynomials with their nonconstant irreducible factors.

Why Lazard?

- Able to handle *curtains* [NDS19] (regions where a polynomial nullifies), which cause other projections to fail.
- Other projections would have to first remove curtains and thus would not technically produce an open CAD.
- So this open CAD is an important first step towards a full CAD.

Open CAD

- **Full CAD:** CAD($\mathcal{F}_n$), composed of j-cells, $0 \leq j \leq n$.
- **Open CAD:** Only contains full-dimensional n-cells.
  - open solution sets of systems of strict polynomial inequalities [Str00].

Why?

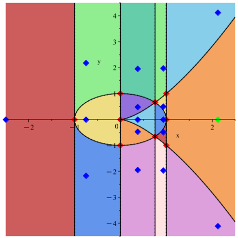
Quicker: Fewer cells produced at each level ($n+1$ vs $2n+1$ for each set of univariate polynomials)

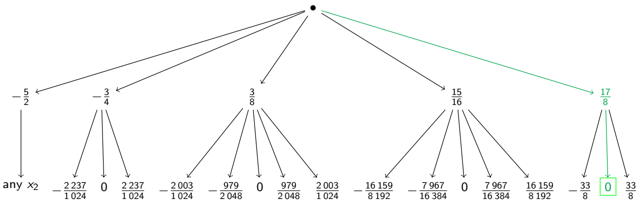
Sufficient: Many real-world applications are interested only in strict inequalities.

Limitation: *Macaulay2* does not currently support polynomial division over $\mathbb{R}$, does not symbolically store roots. Open CAD is possible over $\mathbb{Q}[x_1, \dots, x_n]$.

The CADecomposition Package for *Macaulay2*

- First implementation of CAD in *M2*.
- Produces tree of sample points in $\mathbb{Q}^n$ representing the open cells of a CAD of $\mathbb{R}^n$ for input polynomials in $\mathbb{Q}[x_1, \ldots, x_n]$.
- Able to: return full tree or branches, check these for points where all polynomials are positive, step through the algorithm.
- Improves/fixes `RealRoots`' real root isolation.





The `CADecomposition` Package for *Macaulay2*

- `findPositiveSolution`
  - `openCAD`
    - `projectionPhase`
      - `gmodsHeuristic`
      - `lazardProjection`
        - `leadCoeff`
        - `factorsInList`
          - `factors`
        - `discriminant`
        - `resultant`
    - `liftingPoint`
      - `evalPolys`
      - `samplePoints`
        - `realRootIsolation`
  - `positivePoint`
    - `evalPolys`
  - `hashify`

$$
\begin{tikzcd}
\mathcal{F}_n \arrow[d, "\text{Proj}"] & \mathcal{S}_n \\
\mathcal{F}_{n-1} \arrow[d, "\text{Proj}"] & \mathcal{S}_{n-1} \arrow[u, "\text{Lift}"'] \\
\vdots \arrow[d, "\text{Proj}"] & \vdots \arrow[u, "\text{Lift}"'] \\
\mathcal{F}_2 \arrow[d, "\text{Proj}"] & \mathcal{S}_2 \arrow[u, "\text{Lift}"'] \\
\mathcal{F}_1 \arrow[r, "\text{find roots}", "\text{make sample points}"'] & \mathcal{S}_1 \arrow[u, "\text{Lift}"']
\end{tikzcd}
$$

Figure: List of functions contained in the `CADecomposition` package. Methods labelled in blue are from external packages.

Real Root Isolation

Existing package `RealRoots` performs real root isolation for base phase:

- Take univariate polynomial f with roots $\mathcal{A}(f) = \{\alpha_1, \ldots, \alpha_q\}$, $\alpha_1 < \alpha_2 < \cdots < \alpha_{q-1} < \alpha_q$, $\alpha_h \in \mathbb{R}$.
- Make it *squarefree*: $\hat{f} = sf(f)$, $\mathcal{A}(\hat{f}) = \mathcal{A}(f)$.
- Produce *Sturm Sequence* for this polynomial.
- Create real root bound M.
- Perform *real root isolation* via interval bisection on $[-M, M)$ to obtain a suitable interval $I_h = [I_{h,0}, I_{h,1})$ for each root α_h such that $I_{h,0}, I_{h,1} \in \mathbb{Q}$, $I_{h,0} \leq \alpha_h \leq I_{h,1}$ and the interval contains exactly one root.
  - $\#$ roots in $[a, b)$ = difference in number of (nonzero) sign changes in Sturm sequence evaluated at a and b.

Real Root Isolation

Some issues with this:

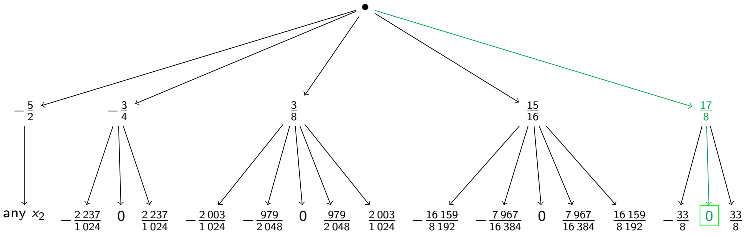
- **Only works on one polynomial:** Our package `CADecomposition` simply performs this real root isolation on $\prod f_i$, where each f_i is reduced to its irreducible factors during the projection phase.
- **Error in root bound:** Original root bound M has an error causing the process to fail when the leading coefficient was negative. This has been fixed.
- **Inefficient root bound:** Original root bound was also weaker than it needed to be, and was incredibly high in cases of coefficient blowup. Fix implements a choice of two root bounds and takes the smaller, speeding up each real root isolation step.
- **Inefficient with close roots:** If two roots were very close, the original algorithm would reduce every isolating interval to the size of the smallest one, creating extra work. Now isolating intervals are not bisected further when complete.
- **Roots not always ordered:** Original algorithm iterates through interval and stops when final one is complete, regardless of where. Fix now lists intervals lowest to highest.

Real Root Isolation

For these intervals $I_h = [I_{h,0}, I_{h,1})$ for each root α_h, we take the open sample points:

$$sp(\mathcal{F}) = \left\{ \frac{I_{h,1} + I_{h+1,0}}{2} \,\middle|\, 1 \leq h \leq q-1 \right\} \cup \{ I_{1,0} - 1,\ I_{q,1} + 1 \},$$

and in the lifting phase these are substituted into $\mathcal{F}_k$, producing a tree of sample points:



Where to Try It

Figure: `https://github.com/cel34-bath/CAD_M2_ICMS_2024`

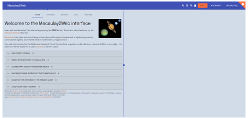


Figure: `https://macaulay2.com/`

Other stuff

- Improve output - currently outputs as a Mutable Hash Table.
- Improve efficiency - small improvements will make big differences.
- More useful outputs: lists of n-tuple s.p.s and signs.
- Showing projection ordering

References I

[Col75] George E. Collins. "Quantifier elimination for real closed fields by cylindrical algebraic decomposition". In: *Lecture Notes in Computer Science* (1975).

[CZC20] Changbo Chen, Zhangpeng Zhu, and Haoyu Chi. "Variable Ordering Selection for Cylindrical Algebraic Decomposition with Artificial Neural Networks". In: July 2020, pp. 281–291.

[dE22] Tereso del Río and Matthew England. "New Heuristic to Choose a Cylindrical Algebraic Decomposition Variable Ordering Motivated by Complexity Analysis". In: *Computer Algebra in Scientific Computing*. Ed. by François Boulier et al. Springer International Publishing, 2022.

[EF19] Matthew England and Dorian Florescu. "Comparing Machine Learning Models to Choose the Variable Ordering for Cylindrical Algebraic Decomposition". In: *Intelligent Computer Mathematics*. Springer International Publishing, 2019, pp. 93–108.

[Hua+15] Zongyan Huang et al. "A Comparison of Three Heuristics to Choose the Variable Ordering for Cylindrical Algebraic Decomposition". In: *ACM Communications in Computer Algebra* 48.3/4 (Feb. 2015), pp. 121–123.

[Laz94] D. Lazard. "An Improved Projection for Cylindrical Algebraic Decomposition". In: *Algebraic Geometry and its Applications: Collections of Papers from Shreeram S. Abhyankar's 60th Birthday Conference*. Springer New York, 1994.

References II

[Man+12] Montserrat Manubens et al. “Cusp Points in the Parameter Space of Degenerate 3-RPR Planar Parallel Manipulators”. In: *Journal of Mechanisms and Robotics* 4.4 (2012).

[MDE18] Casey B. Mulligan, James H. Davenport, and Matthew England. “TheoryGuru: A Mathematica Package to Apply Quantifier Elimination Technology to Economics”. In: *Mathematical Software – ICMS 2018*. Ed. by James H. Davenport et al. Springer International Publishing, 2018.

[MPP19] Scott McCallum, Adam Parusiński, and Laurentiu Paunescu. “Validity proof of Lazard’s method for CAD construction”. In: *Journal of Symbolic Computation* 92 (2019), pp. 52–69.

[NDS19] Akshar Sajive Nair, James Davenport, and Gregory Sankaran. “On Benefits of Equality Constraints in Lex-Least Invariant CAD (Extended Abstract”. In: *Proceedings SC2 2019*. 2019.

[RS21] Gergely Röst and AmirHosein Sadeghimanesh. “Exotic Bifurcations in Three Connected Populations with Allee Effect”. In: *International Journal of Bifurcation and Chaos* 31.13 (2021).

[Str00] Adam Strzeboński. “Solving Systems of Strict Polynomial Inequalities”. In: *Journal of Symbolic Computation* 29.3 (2000).

The End

Figure: https://github.com/cel34-bath/CAD_M2_ICMS_2024