CAD: Algorithmic Real Algebraic Geometry

Zak Tonks[1] [2]
University of Bath
z.p.tonks@bath.ac.uk

20 June 2018

[1]Many thanks to my supervisor James Davenport, and colleagues Akshar Nair (Bath) & Matthew England (Coventry)

[2]Also thanks to Maplesoft, and grants EPSRC EP/J003247/1, EU H2020-FETOPEN-2016-2017-CSA project $\mathcal{SC}^2$ (712689)

Cylindrical Algebraic Decomposition (CAD) is an algorithm used to tackle several problems in real algebraic geometry, such as

Cylindrical Algebraic Decomposition (CAD) is an algorithm used to tackle several problems in real algebraic geometry, such as

- Quantifier Elimination (QE) over the reals,

Cylindrical Algebraic Decomposition (CAD) is an algorithm used to tackle several problems in real algebraic geometry, such as

- Quantifier Elimination (QE) over the reals,
- motion planning in robotics,

Cylindrical Algebraic Decomposition

Cylindrical Algebraic Decomposition (CAD) is an algorithm used to tackle several problems in real algebraic geometry, such as

- Quantifier Elimination (QE) over the reals,
- motion planning in robotics,
- "piano mover's problem"

Cylindrical Algebraic Decomposition

Cylindrical Algebraic Decomposition (CAD) is an algorithm used to tackle several problems in real algebraic geometry, such as

- Quantifier Elimination (QE) over the reals,
- motion planning in robotics,
- "piano mover's problem"

I'll focus on QE over the reals.

Cylindrical Algebraic Decomposition

Problem (Quantifier Elimination)

Given a quantified statement about polynomials $f_i \in \mathbf{Q}[x_1, \ldots, x_n]$

$$\Phi_j := Q_{j+1}x_{j+1} \cdots Q_n x_n \Phi(f_i) \qquad Q_i \in \{\forall, \exists\} \tag{1}$$

produce an equivalent $\Psi(g_i) : g_i \in \mathbf{Q}[x_1, \ldots, x_j]$*: "equivalent"* $\equiv$ *"same real solutions".*

Cylindrical Algebraic Decomposition

Problem (Quantifier Elimination)

Given a quantified statement about polynomials $f_i \in \mathbf{Q}[x_1, \ldots, x_n]$

$$\Phi_j := Q_{j+1}x_{j+1} \cdots Q_n x_n \Phi(f_i) \qquad Q_i \in \{\forall, \exists\} \tag{1}$$

produce an equivalent $\Psi(g_i) : g_i \in \mathbf{Q}[x_1, \ldots, x_j]$*: "equivalent"* $\equiv$ *"same real solutions".*

Solution [Col75]: produce a Cylindrical Algebraic Decomposition of $\mathbf{R}^n$ such that each f_i is sign-invariant on each cell, and the cells are *cylindrical*: $\forall i, \alpha, \beta$ the projections $P_{x_1,\ldots,x_i}(C_\alpha)$ and $P_{x_1,\ldots,x_i}(C_\beta)$ are equal or disjoint.

Problem (Quantifier Elimination)

Given a quantified statement about polynomials $f_i \in \mathbf{Q}[x_1, \ldots, x_n]$

$$\Phi_j := Q_{j+1}x_{j+1} \cdots Q_n x_n \Phi(f_i) \qquad Q_i \in \{\forall, \exists\} \tag{1}$$

produce an equivalent $\Psi(g_i) : g_i \in \mathbf{Q}[x_1, \ldots, x_j]$*: "equivalent"* $\equiv$ *"same real solutions".*

Solution [Col75]: produce a Cylindrical Algebraic Decomposition of $\mathbf{R}^n$ such that each f_i is sign-invariant on each cell, and the cells are *cylindrical*: $\forall i, \alpha, \beta$ the projections $P_{x_1,\ldots,x_i}(C_\alpha)$ and $P_{x_1,\ldots,x_i}(C_\beta)$ are equal or disjoint. Each cell has a sample point s_i (normally arranged cylindrically) and then the truth of Φ in a cell is the truth at a sample point, and $\forall x_r$ becomes $\bigwedge_{x_r \text{ samples}}$ etc.

An example

Consider the problem $\exists y \exists x \; x^2 + y^2 < 1 \wedge 2x < -1$.

An example

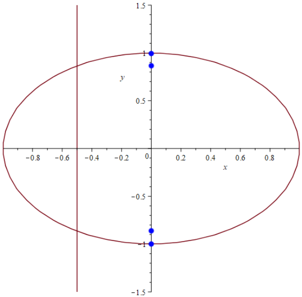
Consider the problem $\exists y \exists x \; x^2 + y^2 < 1 \wedge 2x < -1$. We give CAD the set $\{2x - 1, x^2 + y^2 - 1\}$, and suppose we project onto the y axis.

An example

Consider the problem $\exists y \exists x\ x^2 + y^2 < 1 \wedge 2x < -1$. We give CAD the set $\{2x-1, x^2+y^2-1\}$, and suppose we project onto the y axis.

The non trivial parts of our projection are

$$\{ \underbrace{4-4y^2}_{discrim_x(x^2+y^2-1)}, \underbrace{4y^2-3}_{res_x(x^2+y^2-1,2x+1)} \}$$



Plus/Minus of CAD

Plus/Minus of CAD

+ Solves the problem given, e.g. $\forall x \exists y \; f > 0 \wedge (g = 0 \vee h < 0)$

Plus/Minus of CAD

+ Solves the problem given, e.g. $\forall x \exists y \; f > 0 \wedge (g = 0 \vee h < 0)$
+ The same structure solves all other problems with the same polynomials and order of quantified variables, e.g. $\forall y \; f = 0 \vee (g < 0 \wedge h > 0)$

Plus/Minus of CAD

- $+$ Solves the problem given, e.g. $\forall x \exists y \; f > 0 \wedge (g = 0 \vee h < 0)$
- $+$ The same structure solves all other problems with the same polynomials and order of quantified variables, e.g. $\forall y \; f = 0 \vee (g < 0 \wedge h > 0)$
- $-$ Current algorithms can be misled by spurious solutions. Consider $\{x^2 + y^2 - 2, (x-6)^2 + y^2 - 2\}$. Because $x = 3, y = \pm\sqrt{-7}$ is a common zero, current algorithms wrongly regard $x = 3$ as a critical point over $\mathbf{R}^2$ (which it would be over $\mathbf{C}^2$).

Plus/Minus of CAD

Plus/Minus of CAD

– Not sensitive to structure - $\wedge/\vee$ are lost in favour of giving CAD every polynomial appearing in the formula

Plus/Minus of CAD

- Not sensitive to structure - $\wedge/\vee$ are lost in favour of giving CAD every polynomial appearing in the formula
- Can work very hard on trivial examples:
  $x < -1 \wedge x > 1 \wedge \underbrace{(f_1(x) > 0 \vee \cdots)}_{\text{irrelevant}}$

Plus/Minus of CAD

– Not sensitive to structure - $\wedge/\vee$ are lost in favour of giving CAD every polynomial appearing in the formula

– Can work very hard on trivial examples:
$x < -1 \wedge x > 1 \wedge \underbrace{(f_1(x) > 0 \vee \cdots)}_{\text{irrelevant}}$

+/– Another technique for QE, "Virtual Term Substitution" revolves around "virtually" substituting the roots of the polynomials appearing in the formula into the whole formula, which is highly sensitive to the formula structure and thus not overkill

Plus/Minus of CAD

- – Not sensitive to structure - $\wedge/\vee$ are lost in favour of giving CAD every polynomial appearing in the formula
- – Can work very hard on trivial examples: $x < -1 \wedge x > 1 \wedge \underbrace{(f_1(x) > 0 \vee \cdots)}_{\text{irrelevant}}$
- +/– Another technique for QE, "Virtual Term Substitution" revolves around "virtually" substituting the roots of the polynomials appearing in the formula into the whole formula, which is highly sensitive to the formula structure and thus not overkill
- But this means the polynomials must be solvable by radicals, and complex roots of cubics and above complicate matters

Plus/Minus of CAD

- $-$ Not sensitive to structure - $\wedge/\vee$ are lost in favour of giving CAD every polynomial appearing in the formula
- $-$ Can work very hard on trivial examples: $x < -1 \wedge x > 1 \wedge \underbrace{(f_1(x) > 0 \vee \cdots)}_{\text{irrelevant}}$
- $+/-$ Another technique for QE, "Virtual Term Substitution" revolves around "virtually" substituting the roots of the polynomials appearing in the formula into the whole formula, which is highly sensitive to the formula structure and thus not overkill
- But this means the polynomials must be solvable by radicals, and complex roots of cubics and above complicate matters

  So only really feasible when the degrees of the polynomials involved are low

The original complexity of CAD

The original complexity of CAD

When Collins [Col75] produced his Cylindrical Algebraic Decomposition algorithm, the complexity was $O\left(d^{2^{2n+8}} m^{2^{n+6}}\right) l^3 k$, where

The original complexity of CAD

When Collins [Col75] produced his Cylindrical Algebraic Decomposition algorithm, the complexity was $O\left(d^{2^{2n+8}} m^{2^{n+6}}\right) l^3 k$, where

- n is the number of variables

The original complexity of CAD

When Collins [Col75] produced his Cylindrical Algebraic Decomposition algorithm, the complexity was $O\left(d^{2^{2n+8}} m^{2^{n+6}}\right) l^3 k$, where

- n is the number of variables
- d the maximum degree of any input polynomial in any variable

The original complexity of CAD

When Collins [Col75] produced his Cylindrical Algebraic Decomposition algorithm, the complexity was $O\left(d^{2^{2n+8}} m^{2^{n+6}}\right) l^3 k$, where

- n is the number of variables
- d the maximum degree of any input polynomial in any variable
- m the number of polynomials occurring in the input

The original complexity of CAD

When Collins [Col75] produced his Cylindrical Algebraic Decomposition algorithm, the complexity was $O\left(d^{2^{2n+8}}m^{2^{n+6}}\right)l^3k$, where

- n is the number of variables
- d the maximum degree of any input polynomial in any variable
- m the number of polynomials occurring in the input
- k the number of occurrences of polynomials (essentially the length)

The original complexity of CAD

When Collins [Col75] produced his Cylindrical Algebraic Decomposition algorithm, the complexity was $O\left(d^{2^{2n+8}} m^{2^{n+6}}\right) l^3 k$, where

- n is the number of variables
- d the maximum degree of any input polynomial in any variable
- m the number of polynomials occurring in the input
- k the number of occurrences of polynomials (essentially the length)
- and l the maximum coefficient length.

The original complexity of CAD

When Collins [Col75] produced his Cylindrical Algebraic Decomposition algorithm, the complexity was $O\left(d^{2^{2n+8}} m^{2^{n+6}}\right) l^3 k$, where

- n is the number of variables
- d the maximum degree of any input polynomial in any variable
- m the number of polynomials occurring in the input
- k the number of occurrences of polynomials (essentially the length)
- and l the maximum coefficient length.

The original complexity of CAD

When Collins [Col75] produced his Cylindrical Algebraic Decomposition algorithm, the complexity was $O\left(d^{2^{2n+8}} m^{2^{n+6}}\right) l^3 k$, where

- n is the number of variables
- d the maximum degree of any input polynomial in any variable
- m the number of polynomials occurring in the input
- k the number of occurrences of polynomials (essentially the length)
- and l the maximum coefficient length.

From now on omit l, k, and assume classical arithmetic.

The original complexity of CAD

Given m polynomials of degree d in x_n, we consider P_C:

The original complexity of CAD

Given m polynomials of degree d in x_n, we consider P_C:

1. $O(md)$ coefficients (degree $\leq d$)

The original complexity of CAD

Given m polynomials of degree d in x_n, we consider P_C:

1. $O(md)$ coefficients (degree $\leq d$)
2. $O(md)$ discriminants and subdiscriminants (degree $\leq 2d^2$)

The original complexity of CAD

Given m polynomials of degree d in x_n, we consider P_C:

1. $O(md)$ coefficients (degree $\leq d$)
2. $O(md)$ discriminants and subdiscriminants (degree $\leq 2d^2$)
3. $O(m^2 d)$ resultants and subresultants (degree $\leq 2d^2$)

The original complexity of CAD

Given m polynomials of degree d in x_n, we consider P_C:

1. $O(md)$ coefficients (degree $\leq d$)
2. $O(md)$ discriminants and subdiscriminants (degree $\leq 2d^2$)
3. $O(m^2 d)$ resultants and subresultants (degree $\leq 2d^2$)

The original complexity of CAD

Given m polynomials of degree d in x_n, we consider P_C:

1. $O(md)$ coefficients (degree $\leq d$)
2. $O(md)$ discriminants and subdiscriminants (degree $\leq 2d^2$)
3. $O(m^2 d)$ resultants and subresultants (degree $\leq 2d^2$)

Then make square-free etc., and repeat.

The original complexity of CAD

Given m polynomials of degree d in x_n, we consider P_C:

1. $O(md)$ coefficients (degree $\leq d$)
2. $O(md)$ discriminants and subdiscriminants (degree $\leq 2d^2$)
3. $O(m^2 d)$ resultants and subresultants (degree $\leq 2d^2$)

Then make square-free etc., and repeat.

$$(m, d) \Rightarrow (m^2 d, 2d^2) \Rightarrow (2m^4 d^4, 8d^4) \Rightarrow (32m^8 d^{12}, 128d^8) \Rightarrow \cdots$$

This feed from d to m causes the $d^{2^{2n+O(1)}}$.

Problem (Square-free Decomposition)

Generally a good idea, and often necessary. But one polynomial of degree d might become $O(\sqrt{d})$ polynomials, but the degree might not reduce. Hence (m, d) gets worse when we "improve" the polynomials.

McCallum's Notational Idea [McC84]

Problem (Square-free Decomposition)

Generally a good idea, and often necessary. But one polynomial of degree d might become $O(\sqrt{d})$ polynomials, but the degree might not reduce. Hence (m, d) gets worse when we "improve" the polynomials.

Say that a set of polynomials is (M, D) if it can be partitioned into $\leq M$ sets, with the sum of the degrees in each set $\leq D$. This *is* preserved under square-free, relatively prime, and even complete factorisation, and behaves well w.r.t. operations.

McCallum's Notational Idea [McC84]

Problem (Square-free Decomposition)

Generally a good idea, and often necessary. But one polynomial of degree d might become $O(\sqrt{d})$ polynomials, but the degree might not reduce. Hence (m, d) gets worse when we "improve" the polynomials.

Say that a set of polynomials is (M, D) if it can be partitioned into $\leq M$ sets, with the sum of the degrees in each set $\leq D$. This *is* preserved under square-free, relatively prime, and even complete factorisation, and behaves well w.r.t. operations.

Proposition

If S is an (M, D) set of polynomials in $(x_1, \ldots, x_n)$, then $\{\mathrm{res}_{x_n}(f_i, f_j) : f_i, f_j \in S\}$ is an $\left(\frac{M(M+1)}{2}, 2D^2\right)$ set,

Why the subresultants? McCallum's solution [McC84]

Essentially because the vanishing of $\mathrm{res}(f, g)$ at $(\alpha_1, \ldots, \alpha_{n-1})$ means that f and g cross above there, but the multiplicity of the crossing is determined by the vanishing of subresultants.

Why the subresultants? McCallum's solution [McC84]

Essentially because the vanishing of $\mathrm{res}(f, g)$ at $(\alpha_1, \ldots, \alpha_{n-1})$ means that f and g cross above there, but the multiplicity of the crossing is determined by the vanishing of subresultants.

Hence we may need the subresultants to determine the finer points of the geometry if the resultant vanishes on a set of positive dimension.

Why the subresultants? McCallum's solution [McC84]

Essentially because the vanishing of $\mathrm{res}(f, g)$ at $(\alpha_1, \ldots, \alpha_{n-1})$ means that f and g cross above there, but the multiplicity of the crossing is determined by the vanishing of subresultants.

Hence we may need the subresultants to determine the finer points of the geometry if the resultant vanishes on a set of positive dimension.

Given (M, D) polynomials in x_n, we consider P_M:

Why the subresultants? McCallum's solution [McC84]

Essentially because the vanishing of $\mathrm{res}(f, g)$ at $(\alpha_1, \ldots, \alpha_{n-1})$ means that f and g cross above there, but the multiplicity of the crossing is determined by the vanishing of subresultants.

Hence we may need the subresultants to determine the finer points of the geometry if the resultant vanishes on a set of positive dimension.

Given (M, D) polynomials in x_n, we consider P_M:

1. (MD, D) coefficients (equally, (M, D^2))

Why the subresultants? McCallum's solution [McC84]

Essentially because the vanishing of $\mathrm{res}(f, g)$ at $(\alpha_1, \ldots, \alpha_{n-1})$ means that f and g cross above there, but the multiplicity of the crossing is determined by the vanishing of subresultants.
Hence we may need the subresultants to determine the finer points of the geometry if the resultant vanishes on a set of positive dimension.
Given (M, D) polynomials in x_n, we consider P_M:

1. (MD, D) coefficients (equally, (M, D^2))
2. $(M, 2D^2)$ discriminants

Why the subresultants? McCallum's solution [McC84]

Essentially because the vanishing of $\mathrm{res}(f, g)$ at $(\alpha_1, \ldots, \alpha_{n-1})$ means that f and g cross above there, but the multiplicity of the crossing is determined by the vanishing of subresultants.
Hence we may need the subresultants to determine the finer points of the geometry if the resultant vanishes on a set of positive dimension.
Given (M, D) polynomials in x_n, we consider P_M:

1. (MD, D) coefficients (equally, (M, D^2))
2. $(M, 2D^2)$ discriminants
3. $(O(M^2), 2D^2)$ resultants

Why the subresultants? McCallum's solution [McC84]

Essentially because the vanishing of $\mathrm{res}(f, g)$ at $(\alpha_1, \ldots, \alpha_{n-1})$ means that f and g cross above there, but the multiplicity of the crossing is determined by the vanishing of subresultants.

Hence we may need the subresultants to determine the finer points of the geometry if the resultant vanishes on a set of positive dimension.

Given (M, D) polynomials in x_n, we consider P_M:

1. (MD, D) coefficients (equally, (M, D^2))
2. $(M, 2D^2)$ discriminants
3. $(O(M^2), 2D^2)$ resultants

$(O(M^2), 2D^2)$ in all (no feed from D to M)

Essentially because the vanishing of $\mathrm{res}(f, g)$ at $(\alpha_1, \ldots, \alpha_{n-1})$ means that f and g cross above there, but the multiplicity of the crossing is determined by the vanishing of subresultants.
Hence we may need the subresultants to determine the finer points of the geometry if the resultant vanishes on a set of positive dimension.
Given (M, D) polynomials in x_n, we consider P_M:

1. (MD, D) coefficients (equally, (M, D^2))
2. $(M, 2D^2)$ discriminants
3. $(O(M^2), 2D^2)$ resultants

$(O(M^2), 2D^2)$ in all (no feed from D to M)

Why the subresultants? McCallum's solution [McC84]

Essentially because the vanishing of $\mathrm{res}(f,g)$ at $(\alpha_1,\ldots,\alpha_{n-1})$ means that f and g cross above there, but the multiplicity of the crossing is determined by the vanishing of subresultants.

Hence we may need the subresultants to determine the finer points of the geometry if the resultant vanishes on a set of positive dimension.

Given (M,D) polynomials in x_n, we consider P_M:

1. (MD,D) coefficients (equally, (M,D^2))
2. $(M,2D^2)$ discriminants
3. $(O(M^2),2D^2)$ resultants

   $(O(M^2),2D^2)$ in all (no feed from D to M)

This works for *order-invariance*, rather than just sign-invariance,

Why the subresultants? McCallum's solution [McC84]

Essentially because the vanishing of $\mathrm{res}(f, g)$ at $(\alpha_1, \ldots, \alpha_{n-1})$ means that f and g cross above there, but the multiplicity of the crossing is determined by the vanishing of subresultants.

Hence we may need the subresultants to determine the finer points of the geometry if the resultant vanishes on a set of positive dimension.

Given (M, D) polynomials in x_n, we consider P_M:

1. (MD, D) coefficients (equally, (M, D^2))
2. $(M, 2D^2)$ discriminants
3. $(O(M^2), 2D^2)$ resultants

   $(O(M^2), 2D^2)$ in all (no feed from D to M)

This works for *order-invariance*, rather than just sign-invariance, as long as no polynomial is identically zero on a set of positive dimension ("well-oriented").

Why the subresultants? McCallum's solution [McC84]

Essentially because the vanishing of $\mathrm{res}(f, g)$ at $(\alpha_1, \ldots, \alpha_{n-1})$ means that f and g cross above there, but the multiplicity of the crossing is determined by the vanishing of subresultants.
Hence we may need the subresultants to determine the finer points of the geometry if the resultant vanishes on a set of positive dimension.
Given (M, D) polynomials in x_n, we consider P_M:

1. (MD, D) coefficients (equally, (M, D^2))
2. $(M, 2D^2)$ discriminants
3. $(O(M^2), 2D^2)$ resultants

$(O(M^2), 2D^2)$ in all (no feed from D to M)

This works for *order-invariance*, rather than just sign-invariance, as long as no polynomial is identically zero on a set of positive dimension ("well-oriented").
Note the curiosity that a stronger result has a better algorithm.

The Lazard projection [Laz94, MPP17]

P_L is very similar to P_M (only needs leading and trailing coefficients).

The Lazard projection [Laz94, MPP17]

P_L is very similar to P_M (only needs leading and trailing coefficients).
What is guaranteed is Lazard-invariance, not order-invariance.

The Lazard projection [Laz94, MPP17]

P_L is very similar to P_M (only needs leading and trailing coefficients).
What is guaranteed is Lazard-invariance, not order-invariance.
Like order-invariance, Lazard-invariance is stronger than sign-invariance.

The Lazard projection [Laz94, MPP17]

- P_L is very similar to P_M (only needs leading and trailing coefficients).
- What is guaranteed is Lazard-invariance, not order-invariance.
- Like order-invariance, Lazard-invariance is stronger than sign-invariance.
- The lifting process is different: if a polynomial is nullified, we divide through by the nullifying multiple (and therefore locally lift w.r.t. a different polynomial). Hence we don't need the well-oriented assumption.

Conclusions and Open Problems

1. The true complexity of quantifier elimination comes from the logical structure, especially alternation of quantifiers.
2. The definition of cylindricity means that the results must be applicable for all quantifier structures (with the variables in the same order).
3. However, while the worst case is very bad, there is a lot that can be done with the end structure.
4. Frequent recent interests involve making CAD procedures dynamic, and optimisations in the presence of equational constraints.

Thanks for listening

Questions?

Bibliography

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.

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.

Bibliography II

S. McCallum.
An Improved Projection Operation for Cylindrical Algebraic Decomposition.
PhD thesis, University of Wisconsin-Madison Computer Science, 1984.

S. McCallum, A. Parusinski, and L. Paunescu.
Validity proof of Lazard's method for CAD construction.
https://arxiv.org/pdf/1607.00264v2.pdf, 2017.