# Enhanced CAD-Based Quantifier Elimination with Multiple Equational Constraints

James Davenport (U. Bath)  Matthew England (Coventry U.)
Scott McCallum (Macquarie U.)

**28th International Workshop on Computer Algebra in Scientific Computation**
**CASC 2026, Bath, UK**

31st August – 4th September 2026

Supported by the UKRI EPSRC DEWCAD Project (grants EP/T015713/1 and EP/T015748/1)

# Overview



We announce two enhancements to Cylindrical Algebraic Decomposition (CAD) based real quantifier elimination (QE), in the case where we have multiple equational constraints (ECs):

1. **Detailed Parametric Outputs:** Provides detailed cell structure and solution counts when the input formula's variables are partitioned into parameters and unknowns.
2. **Efficiency Gains:** Validates Collins' fully-reduced equational projection operator beyond the first step, under specific conditions.

The CASC 2026 Extended Abstract summarises a full work-in-progress paper, available as a preprint on the arXiv: `https://doi.org/10.48550/arXiv.2604.23873`

# Outline

# Outline

# Real Quantifier Elimination



## Real Quantifier Elimination (Real QE)

**Given:** A quantified formula (in prenex normal form) whose atoms are (integral) polynomial constraints;
**Produce:** a quantifier-free formula logically equivalent over $\mathbb{R}$.

Fully quantified examples:

Input: $\exists x, x^2 + 3x + 1 > 0$

Output: `True` e.g. when $x = 0$

Input: $\forall x, x^2 + 3x + 1 > 0$

Output: `False` e.g. when $x = -1$

Input: $\forall x, x^2 + 1 > 0$

Output: `True`

Partially quantified example:

Input: $\forall x, x^2 + bx + 1 > 0$

Output: ???

The answer depends on the free (unquantified) variable, $b$.

# Real Quantifier Elimination



## Real Quantifier Elimination (Real QE)

**Given:** A quantified formula (in prenex normal form) whose atoms are (integral) polynomial constraints;
**Produce:** a quantifier-free formula logically equivalent over $\mathbb{R}$.

Fully quantified examples:

Input: $\exists x, x^2 + 3x + 1 > 0$

Output: `True` e.g. when $x = 0$

Input: $\forall x, x^2 + 3x + 1 > 0$

Output: `False` e.g. when $x = -1$

Input: $\forall x, x^2 + 1 > 0$

Output: `True`

Partially quantified example:

Input: $\forall x, x^2 + bx + 1 > 0$

Output: ???

The answer depends on the free (unquantified) variable, $b$.

# Partially Quantified QE Example



Consider $\forall x, x^2 + bx + 1 > 0$.
When $b = 0$ and $b = 3$:

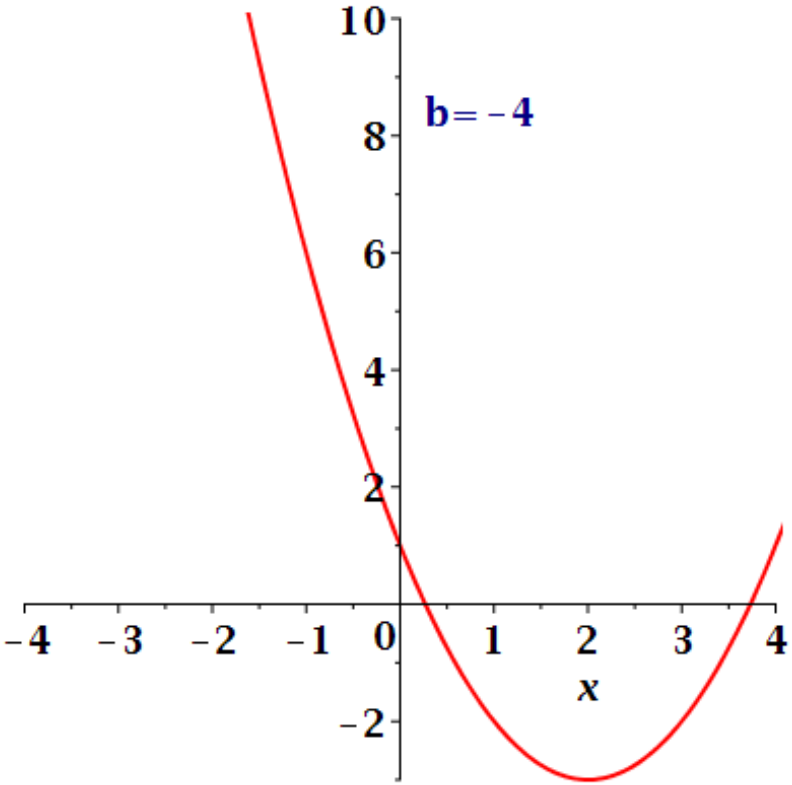
$$\forall x, x^2 + 1 > 0 \text{ is } \texttt{True},$$
$$\forall x, x^2 + 3x + 1 > 0 \text{ is } \texttt{False}.$$

So in general the answer depends on $b$.

Input: $\forall x, x^2 + bx + 1 > 0$

Output: $(-2 < b) \wedge (b < 2)$

Computer algebra technology can produce such answers based on symbolic reasoning.



# Partially Quantified QE Example



Consider $\forall x, x^2 + bx + 1 > 0$.
When $b = 0$ and $b = 3$:

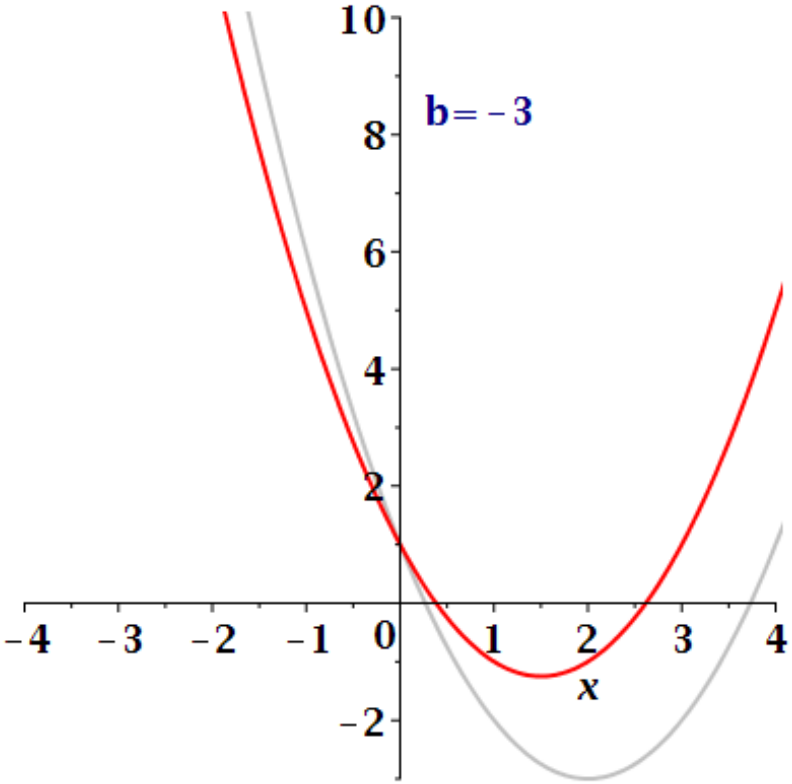
$$\forall x, x^2 + 1 > 0 \text{ is } \texttt{True},$$
$$\forall x, x^2 + 3x + 1 > 0 \text{ is } \texttt{False}.$$

So in general the answer depends on $b$.

Input: $\forall x, x^2 + bx + 1 > 0$

Output: $(-2 < b) \wedge (b < 2)$

Computer algebra technology can produce such answers based on symbolic reasoning.



# Partially Quantified QE Example



Consider $\forall x, x^2 + bx + 1 > 0$.
When $b = 0$ and $b = 3$:

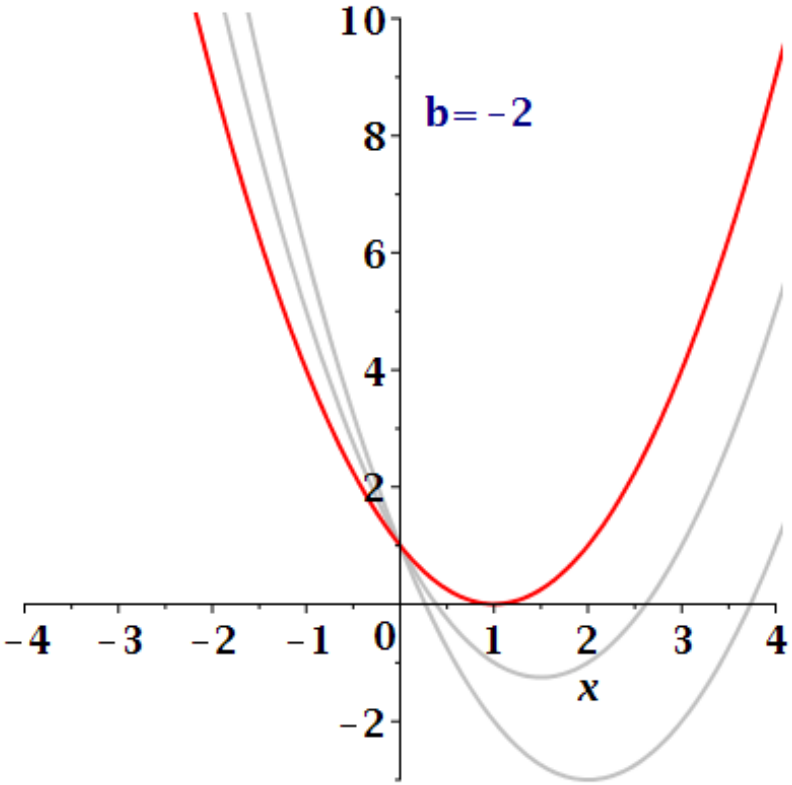
$$\forall x, x^2 + 1 > 0 \text{ is } \texttt{True},$$
$$\forall x, x^2 + 3x + 1 > 0 \text{ is } \texttt{False}.$$

So in general the answer depends on $b$.

Input: $\forall x, x^2 + bx + 1 > 0$

Output: $(-2 < b) \wedge (b < 2)$

Computer algebra technology can produce such answers based on symbolic reasoning.



# Partially Quantified QE Example



Consider $\forall x, x^2 + bx + 1 > 0$.
When $b = 0$ and $b = 3$:

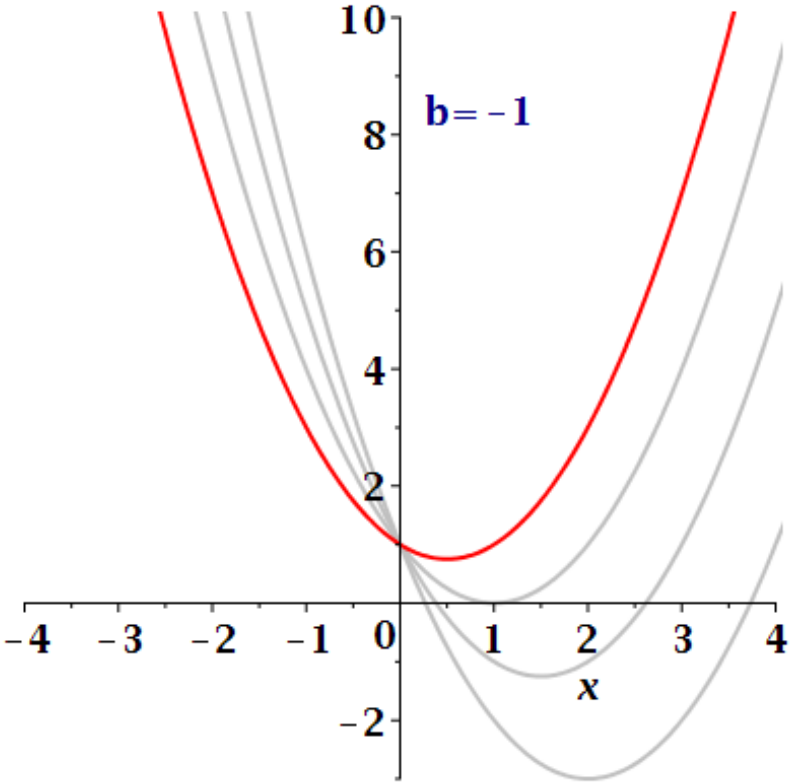
$$\forall x, x^2 + 1 > 0 \text{ is } \texttt{True},$$
$$\forall x, x^2 + 3x + 1 > 0 \text{ is } \texttt{False}.$$

So in general the answer depends on $b$.

Input: $\forall x, x^2 + bx + 1 > 0$

Output: $(-2 < b) \wedge (b < 2)$

Computer algebra technology can produce such answers based on symbolic reasoning.



# Partially Quantified QE Example



Consider $\forall x, x^2 + bx + 1 > 0$.
When $b = 0$ and $b = 3$:

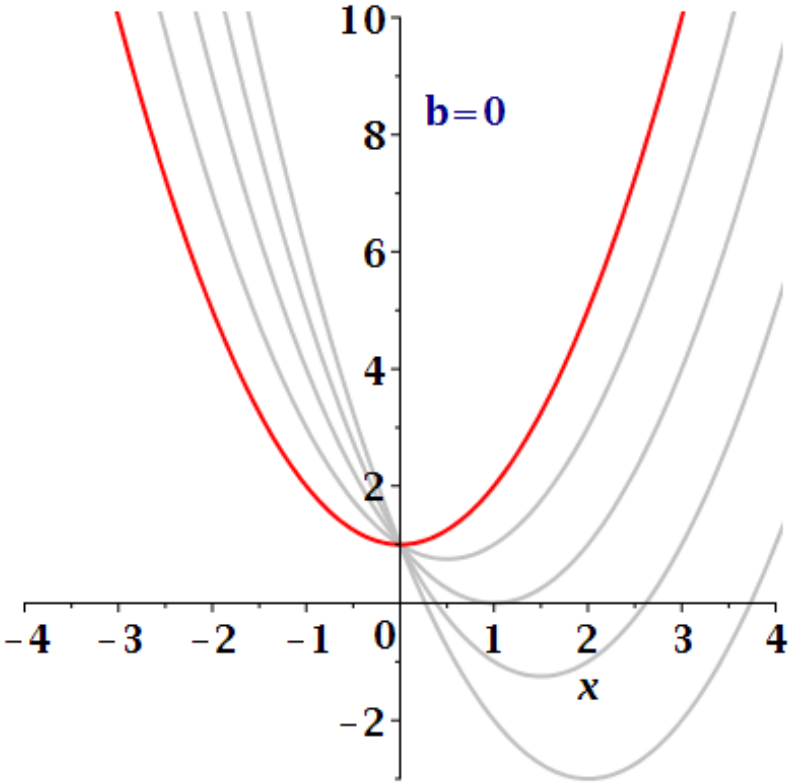
$$\forall x, x^2 + 1 > 0 \text{ is } \texttt{True},$$
$$\forall x, x^2 + 3x + 1 > 0 \text{ is } \texttt{False}.$$

So in general the answer depends on $b$.

Input: $\forall x, x^2 + bx + 1 > 0$

Output: $(-2 < b) \wedge (b < 2)$

Computer algebra technology can produce such answers based on symbolic reasoning.



# Partially Quantified QE Example



Consider $\forall x, x^2 + bx + 1 > 0$.
When $b = 0$ and $b = 3$:

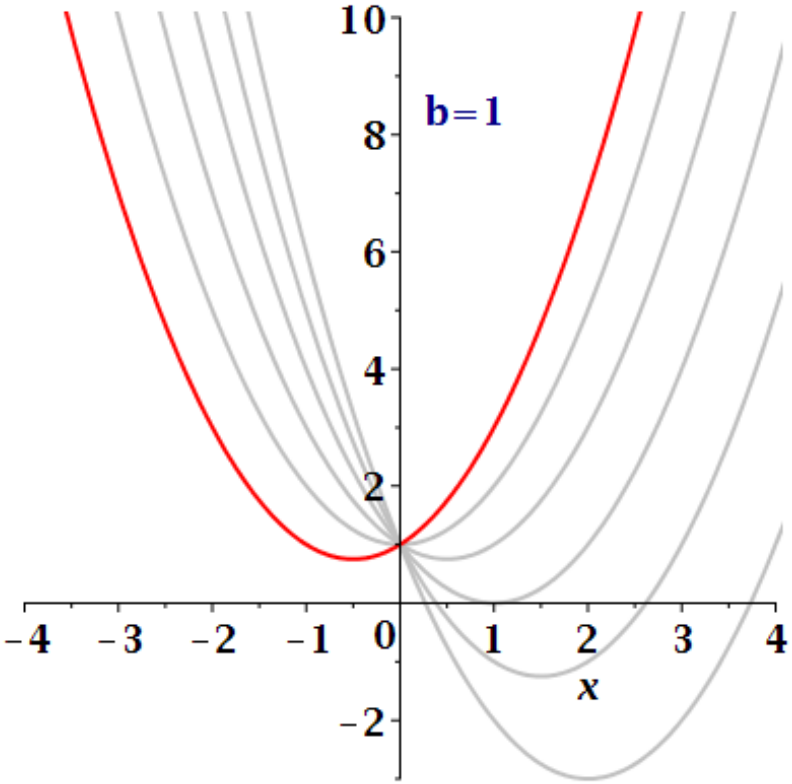
$$\forall x, x^2 + 1 > 0 \text{ is } \texttt{True},$$
$$\forall x, x^2 + 3x + 1 > 0 \text{ is } \texttt{False}.$$

So in general the answer depends on $b$.

Input: $\forall x, x^2 + bx + 1 > 0$

Output: $(-2 < b) \wedge (b < 2)$

Computer algebra technology can produce such answers based on symbolic reasoning.



# Partially Quantified QE Example



Consider $\forall x, x^2 + bx + 1 > 0$.
When $b = 0$ and $b = 3$:

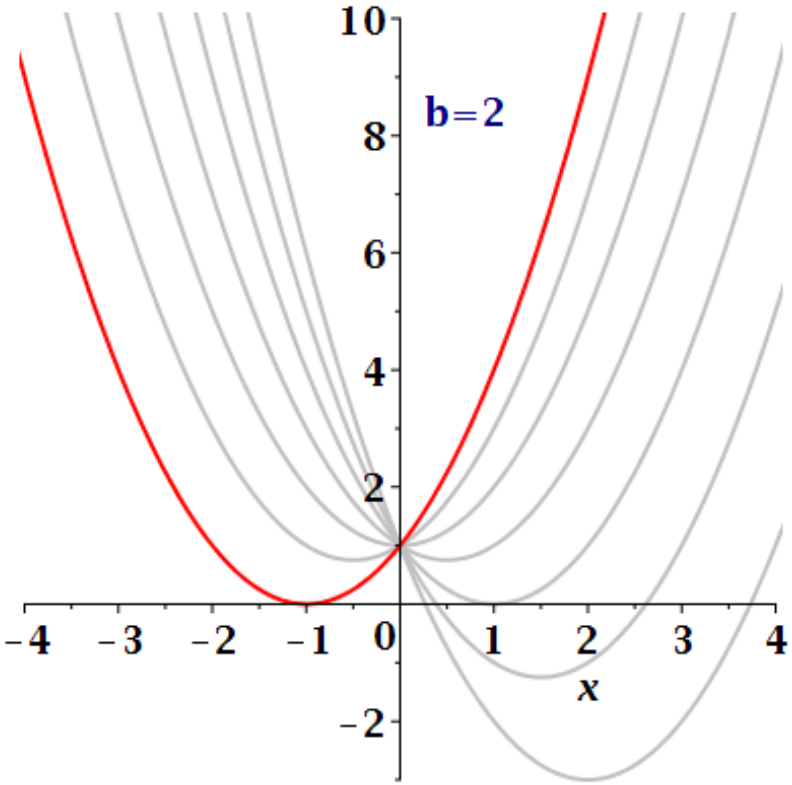
$$\forall x, x^2 + 1 > 0 \text{ is } \texttt{True},$$
$$\forall x, x^2 + 3x + 1 > 0 \text{ is } \texttt{False}.$$

So in general the answer depends on $b$.

Input: $\forall x, x^2 + bx + 1 > 0$

Output: $(-2 < b) \wedge (b < 2)$

Computer algebra technology can produce such answers based on symbolic reasoning.



# Partially Quantified QE Example



Consider $\forall x, x^2 + bx + 1 > 0$.
When $b = 0$ and $b = 3$:

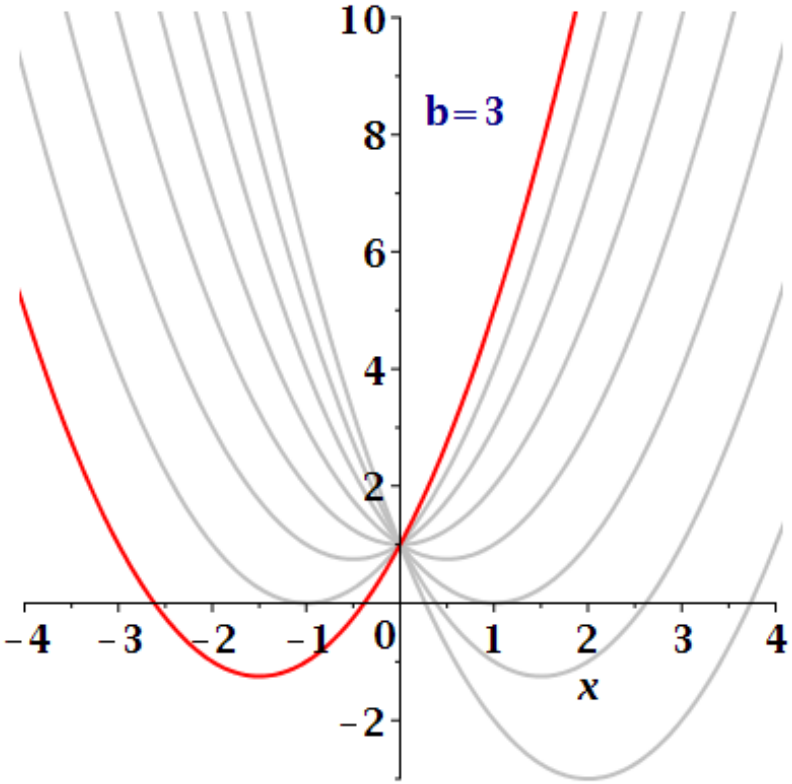
$$\forall x, x^2 + 1 > 0 \text{ is } \texttt{True},$$
$$\forall x, x^2 + 3x + 1 > 0 \text{ is } \texttt{False}.$$

So in general the answer depends on $b$.

Input: $\forall x, x^2 + bx + 1 > 0$

Output: $(-2 < b) \wedge (b < 2)$

Computer algebra technology can produce such answers based on symbolic reasoning.



# Partially Quantified QE Example



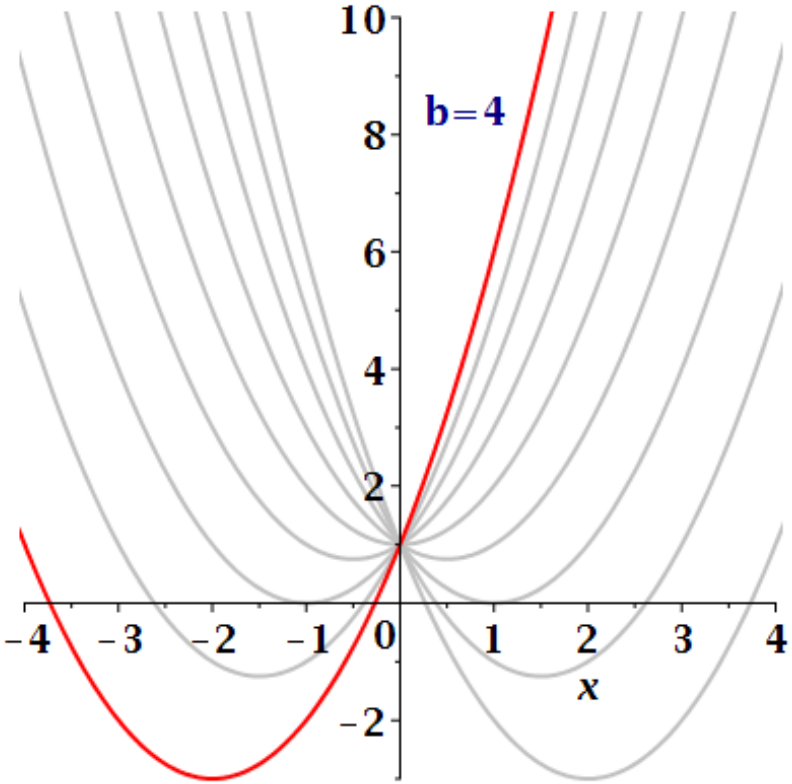
Consider $\forall x, x^2 + bx + 1 > 0$.
When $b = 0$ and $b = 3$:

$$\forall x, x^2 + 1 > 0 \text{ is } \texttt{True},$$
$$\forall x, x^2 + 3x + 1 > 0 \text{ is } \texttt{False}.$$

So in general the answer depends on $b$.

$$\text{Input: } \forall x, x^2 + bx + 1 > 0$$
$$\text{Output: } (-2 < b) \wedge (b < 2)$$

Computer algebra technology can produce such answers based on symbolic reasoning.



# Outline

# Cylindrical Algebraic Decomposition (Slide 4/28)

The main algebraic tool to perform real QE is **Cylindrical Algebraic Decomposition (CAD)**. This:

- Decomposes the variable space into finitely many cells according to the truth of the polynomial constraints involved (but is often much finer than needed). Thus, we can work with one sample point per cell.
- Provides an algebraic formula for each cell.
- Has cells arranged in cylinders, making projection easy.

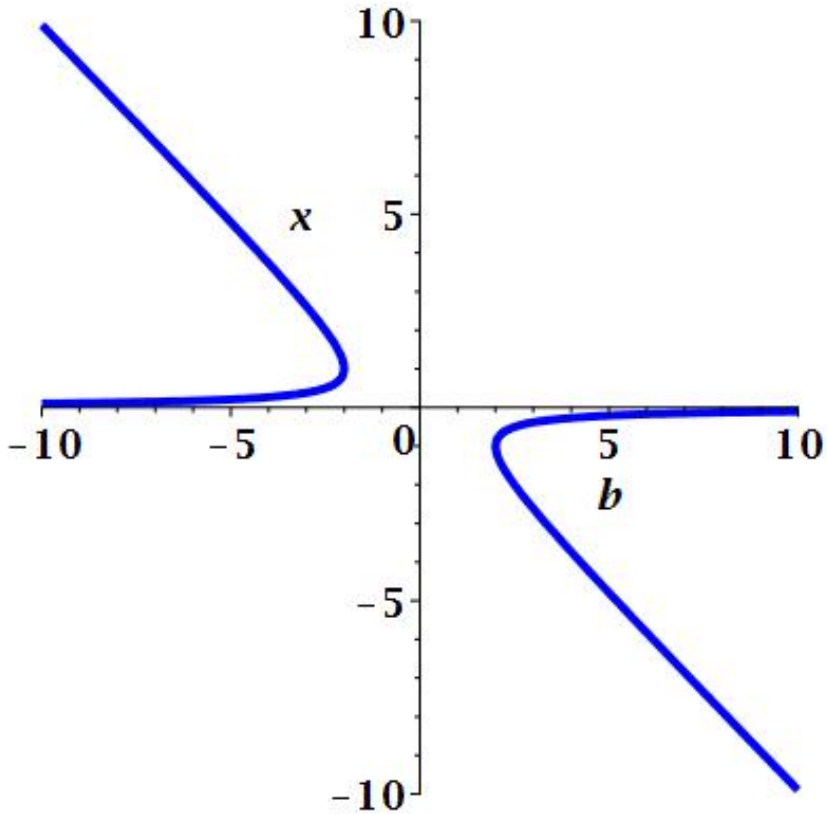
G.E. Collins.
*Quantifier elimination for real closed fields by cylindrical algebraic decomposition.*
In Proc. 2nd GI Conf. Automata Theory and Formal Languages, pp. 134–183. Springer-Verlag, 1975.

# QE via CAD Example



How to determine with CAD?

$$\exists x, x^2 + bx + 1 \leq 0$$



# QE via CAD Example

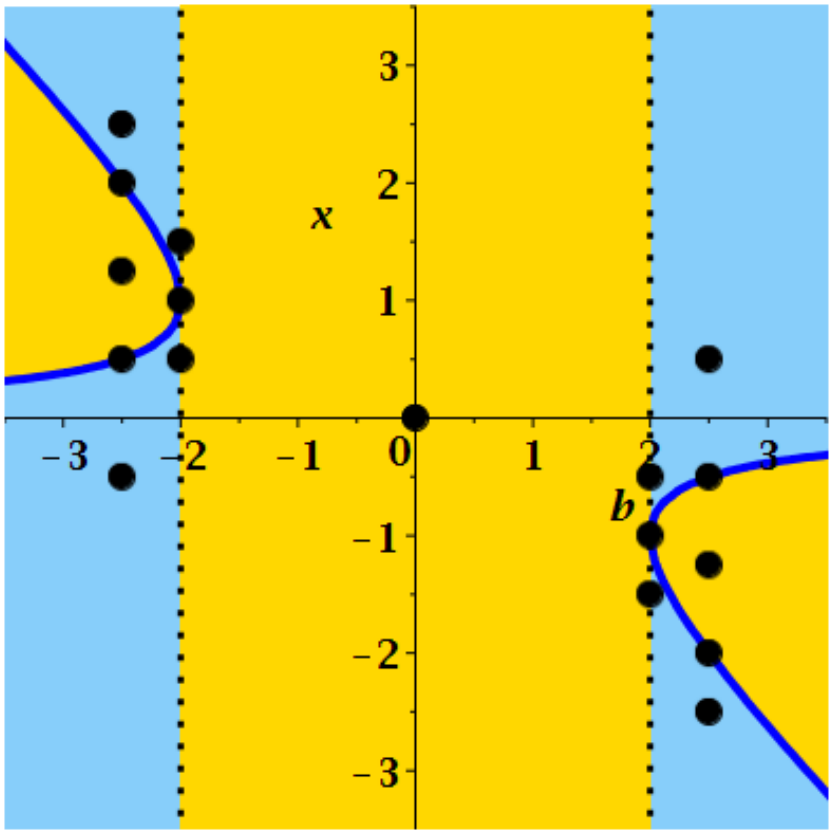


How to determine with CAD?

$$\exists x, x^2 + bx + 1 \leq 0$$

To solve we:

Build a sign-invariant CAD for $f = x^2 + bx + 1$.



# QE via CAD Example



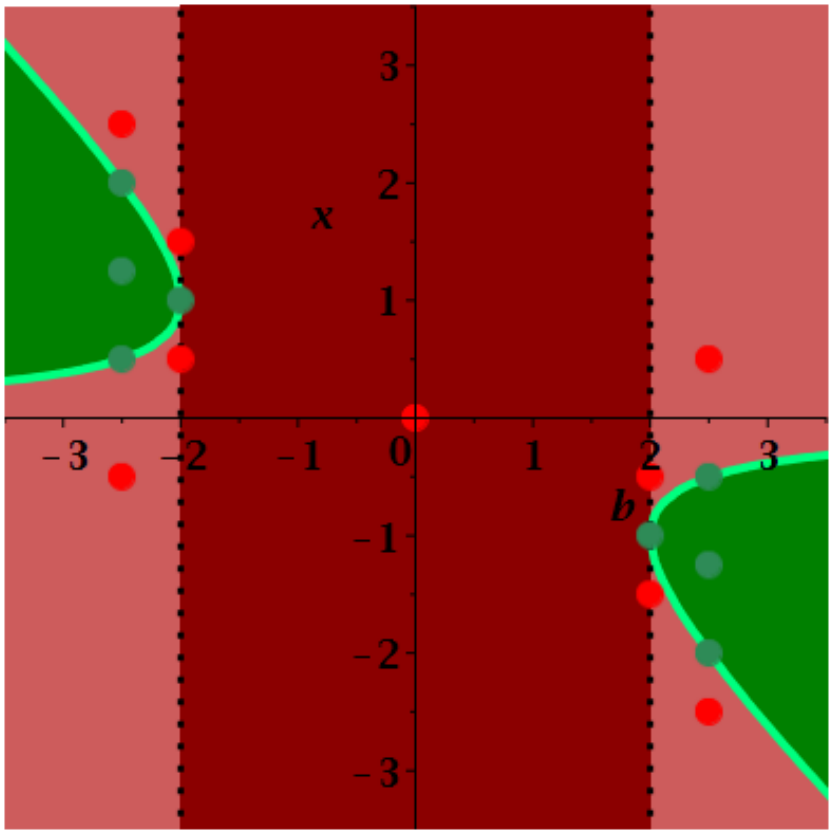
How to determine with CAD?

$$\exists x, x^2 + bx + 1 \leq 0$$

To solve we:

Build a sign-invariant CAD for $f = x^2 + bx + 1$.

Tag each cell true or false according to whether $f \leq 0$.



# QE via CAD Example



How to determine with CAD?

$$\exists x, x^2 + bx + 1 \leq 0$$

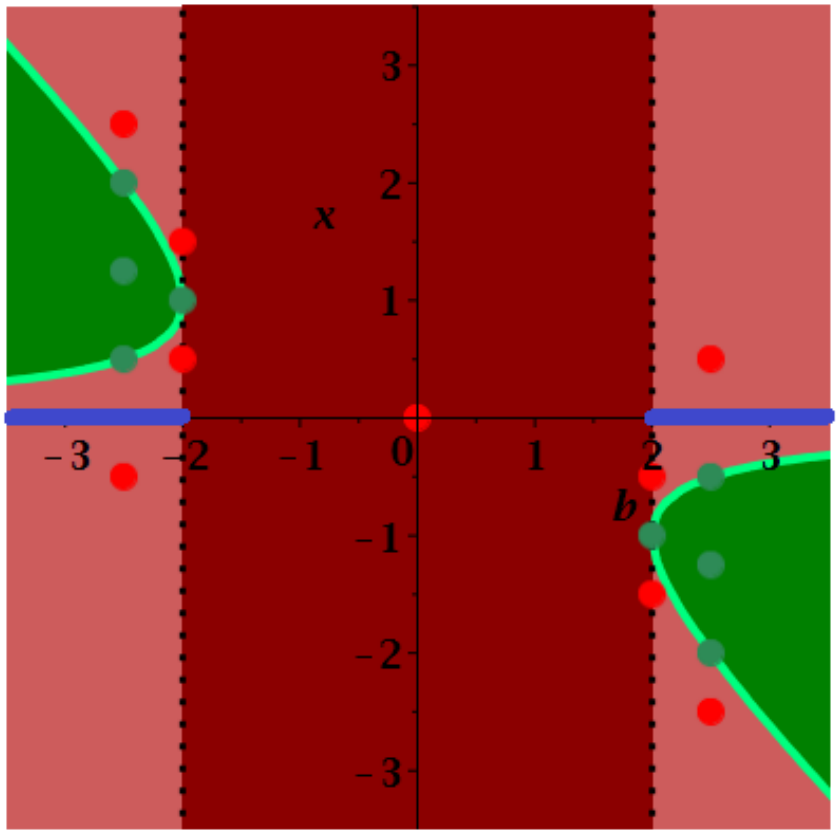
To solve we:

Build a sign-invariant CAD for $f = x^2 + bx + 1$.

Tag each cell true or false according to whether $f \leq 0$.

Take disjunction of projections of true cells:

$$b < -2 \vee b = -2 \vee b = 2 \vee b > 2$$



# QE via CAD Example



How to determine with CAD?

$$\exists x, x^2 + bx + 1 \leq 0$$

To solve we:

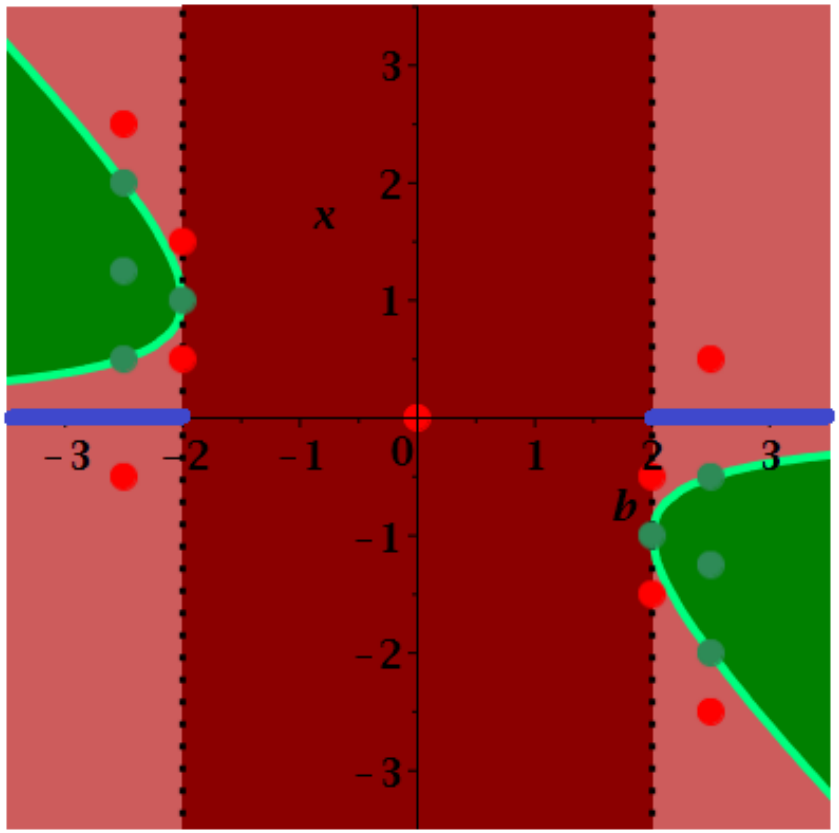
Build a sign-invariant CAD for $f = x^2 + bx + 1$.

Tag each cell true or false according to whether $f \leq 0$.

Take disjunction of projections of true cells:

$$\Longrightarrow$$

$$b \leq -2 \vee b \geq 2$$



# QE via CAD in General



In general we can perform Real QE via CAD as follows.

- Eliminate existential quantifiers by projecting the true cells.
- Convert universal quantifiers to existential by using

$$\forall x P(x) = \neg \exists x \neg P(x)$$

and then proceeding with existential and negating the answer.

Recall our original example was $\forall x, x^2 + bx + 1 > 0$.

- The above leads us to study $\exists x, x^2 + bx + 1 \leq 0$.
- From previous slide this has solution $b \leq -2 \vee b \geq +2$.
- Negate for solution to original problem: $-2 < b \wedge b < +2$
- This is what we found numerically earlier.

**Note:** Projection and negation are both easy with cylindrical cells!

# QE via CAD in General



In general we can perform Real QE via CAD as follows.

- Eliminate existential quantifiers by projecting the true cells.
- Convert universal quantifiers to existential by using

$$\forall x P(x) = \neg \exists x \neg P(x)$$

and then proceeding with existential and negating the answer.

Recall our original example was $\forall x, x^2 + bx + 1 > 0$.

- The above leads us to study $\exists x, x^2 + bx + 1 \leq 0$.
- From previous slide this has solution $b \leq -2 \vee b \geq +2$.
- Negate for solution to original problem: $-2 < b \wedge b < +2$
- This is what we found numerically earlier.

**Note:** Projection and negation are both easy with cylindrical cells!

# CAD Complexity



J.H. Davenport and J. Heintz.
*Real quantifier elimination is doubly exponential.*
Journal of Symbolic Computation, 5(1-2):29–35, 1988

CAD has complexity doubly exponential in the worst case.

C. Brown and J.H. Davenport.
*The complexity of quantifier elimination and cylindrical algebraic decomposition.*
In Proc. ISSAC '07, pages 54–60. ACM, 2007.

By the end of projection we have $M$ polynomials in $\mathbb{R}^1$, each of degree $D$, where $D = d^{2^{O(n)}}$ and $M = m^{2^{O(n)}}$. There are also lower bounds with $D = d^{2^{\Omega(n)}}$ and $M = m^{2^{\Omega(n)}}$. The difficulty of CAD resides in the complicated number of ways simple polynomials can interact.

# CAD Complexity



J.H. Davenport and J. Heintz.
*Real quantifier elimination is doubly exponential.*
Journal of Symbolic Computation, 5(1-2):29–35, 1988

CAD has complexity doubly exponential in the worst case.

C. Brown and J.H. Davenport.
*The complexity of quantifier elimination and cylindrical algebraic decomposition.*
In Proc. ISSAC '07, pages 54–60. ACM, 2007.

By the end of projection we have $M$ polynomials in $\mathbb{R}^1$, each of degree $D$, where $D = d^{2^{O(n)}}$ and $M = m^{2^{O(n)}}$. There are also lower bounds with $D = d^{2^{\Omega(n)}}$ and $M = m^{2^{\Omega(n)}}$. The difficulty of CAD resides in the complicated number of ways simple polynomials can interact.

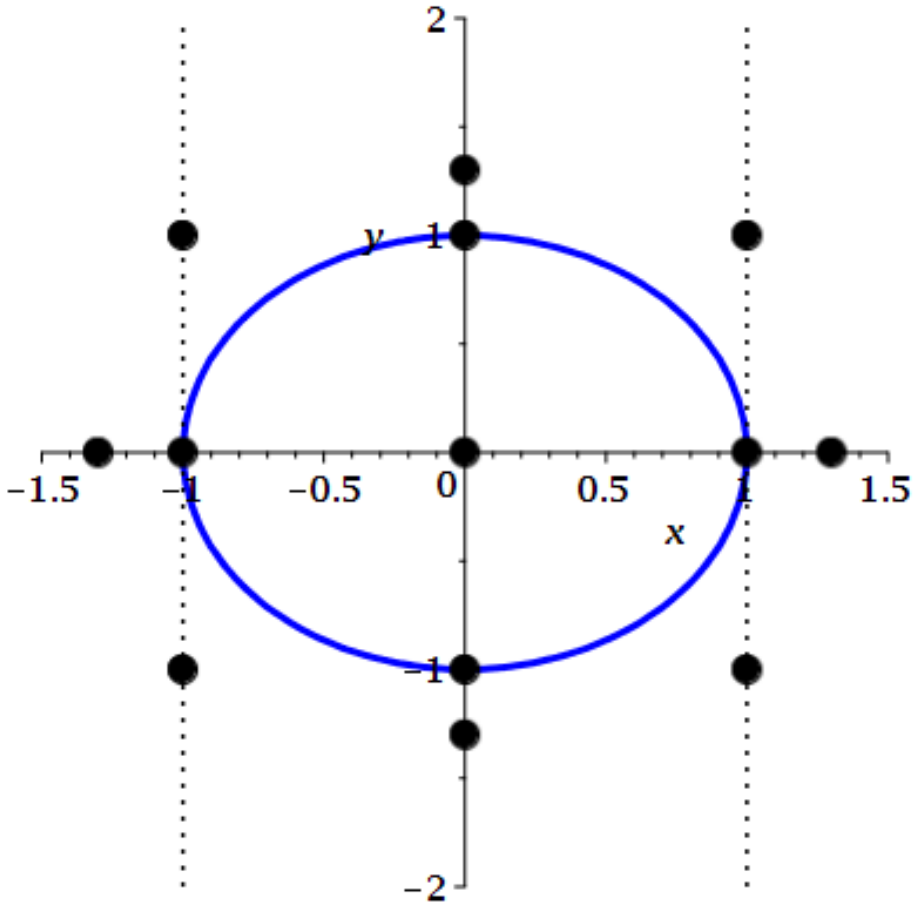
# Outline

# Sign-invariance



Traditionally a CAD is produced from a set of polynomials such that each polynomial has constant sign (positive, zero or negative) in each cell. Such a CAD is said to be **sign-invariant**.

E.g. A CAD produced to be sign-invariant for the polynomial $f := x^2 + y^2 - 1$ will commonly have 13 cells.

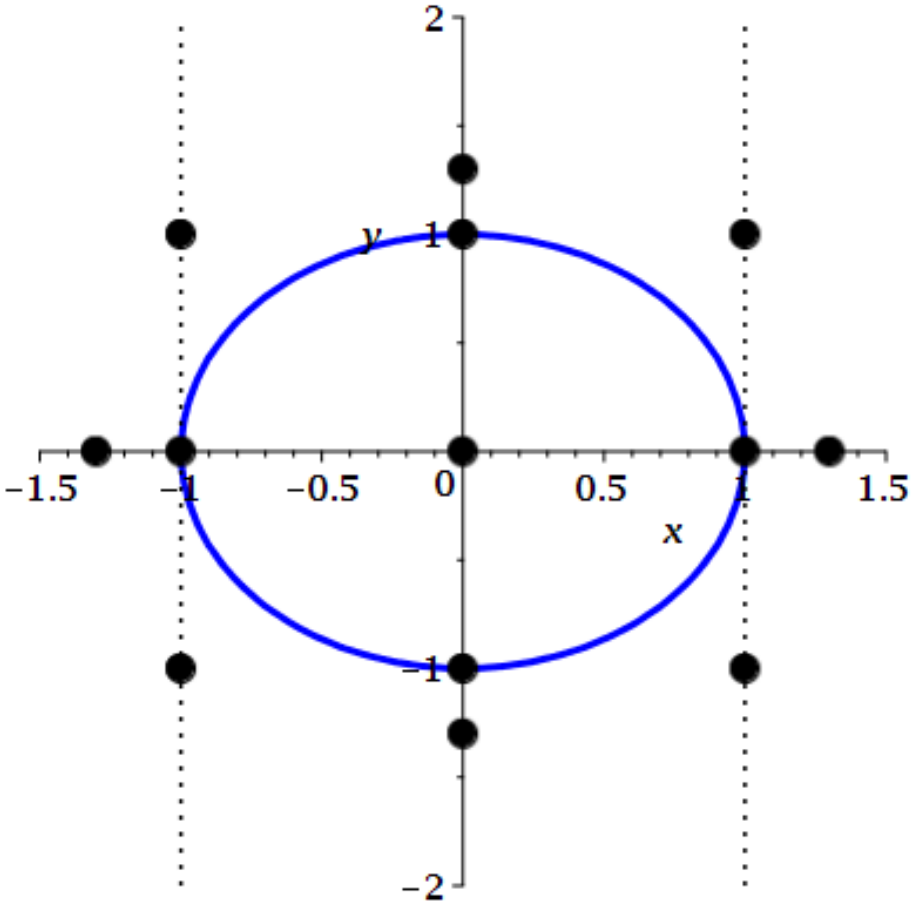


# Sign-invariance



Traditionally a CAD is produced from a set of polynomials such that each polynomial has constant sign (positive, zero or negative) in each cell. Such a CAD is said to be **sign-invariant**.

E.g. A CAD produced to be sign-invariant for the polynomial $f := x^2 + y^2 - 1$ will commonly have 13 cells.

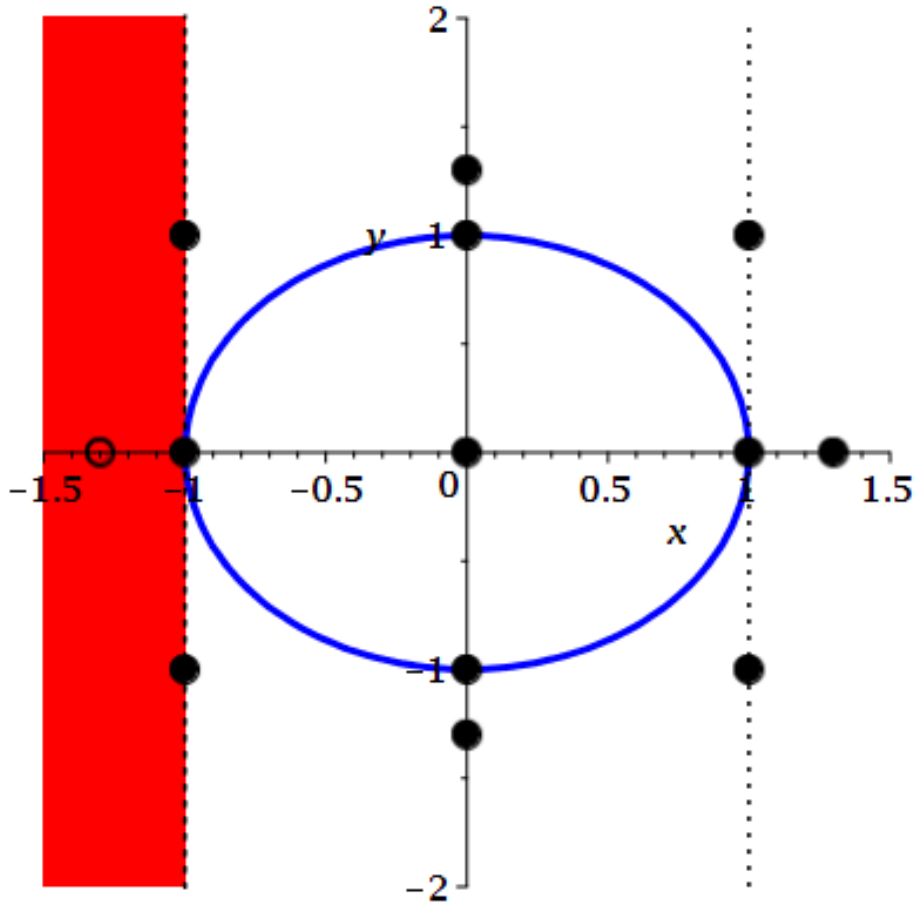


# Sign-invariance



Traditionally a CAD is produced from a set of polynomials such that each polynomial has constant sign (positive, zero or negative) in each cell. Such a CAD is said to be **sign-invariant**.

E.g. A CAD produced to be sign-invariant for the polynomial $f := x^2 + y^2 - 1$ will commonly have 13 cells.

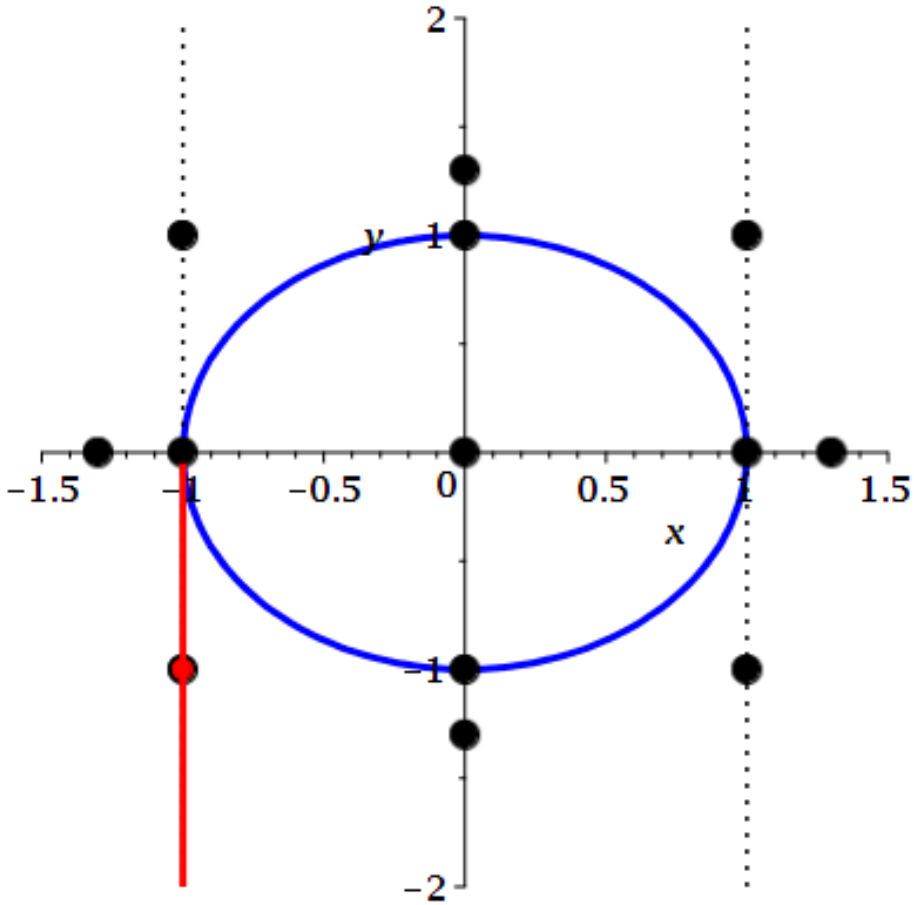


# Sign-invariance



Traditionally a CAD is produced from a set of polynomials such that each polynomial has constant sign (positive, zero or negative) in each cell. Such a CAD is said to be **sign-invariant**.

E.g. A CAD produced to be sign-invariant for the polynomial $f := x^2 + y^2 - 1$ will commonly have 13 cells.

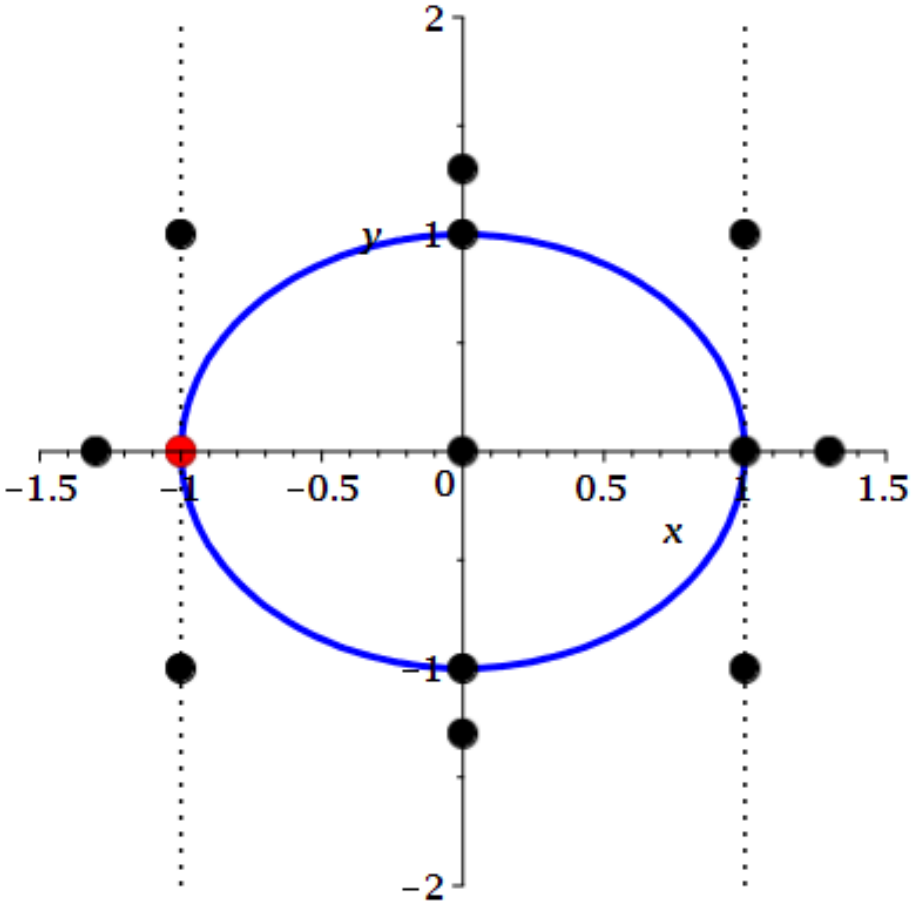


# Sign-invariance



Traditionally a CAD is produced from a set of polynomials such that each polynomial has constant sign (positive, zero or negative) in each cell. Such a CAD is said to be **sign-invariant**.

E.g. A CAD produced to be sign-invariant for the polynomial $f := x^2 + y^2 - 1$ will commonly have 13 cells.

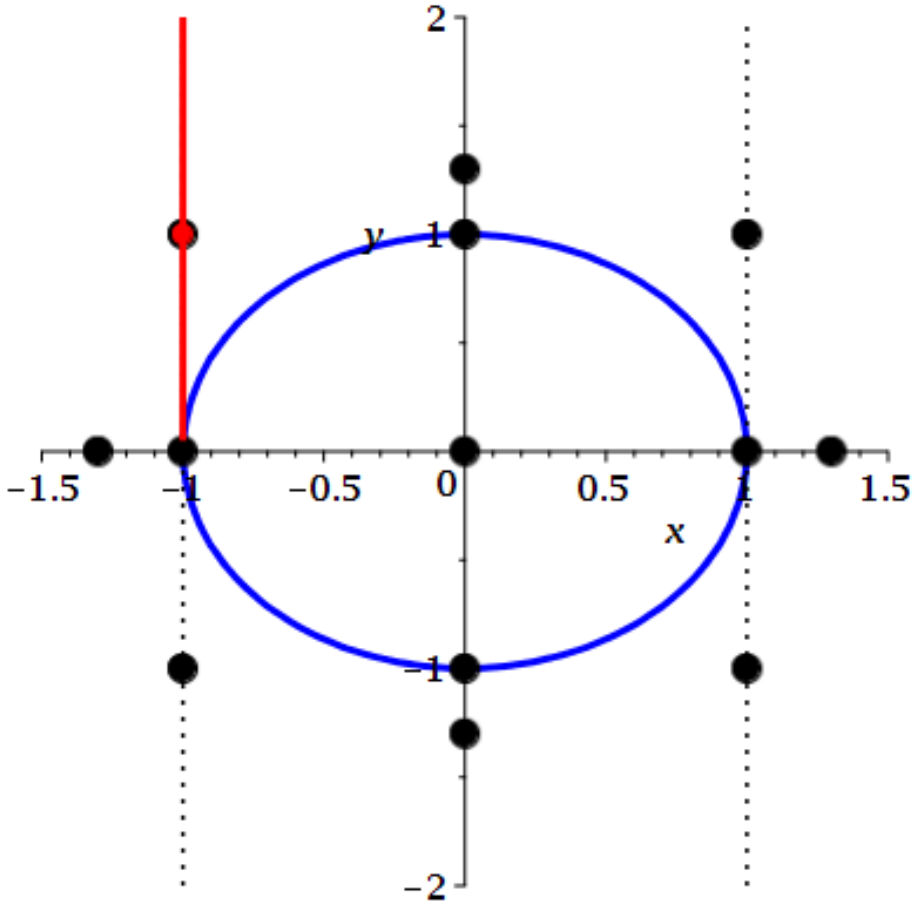


# Sign-invariance



Traditionally a CAD is produced from a set of polynomials such that each polynomial has constant sign (positive, zero or negative) in each cell. Such a CAD is said to be **sign-invariant**.

E.g. A CAD produced to be sign-invariant for the polynomial $f := x^2 + y^2 - 1$ will commonly have 13 cells.

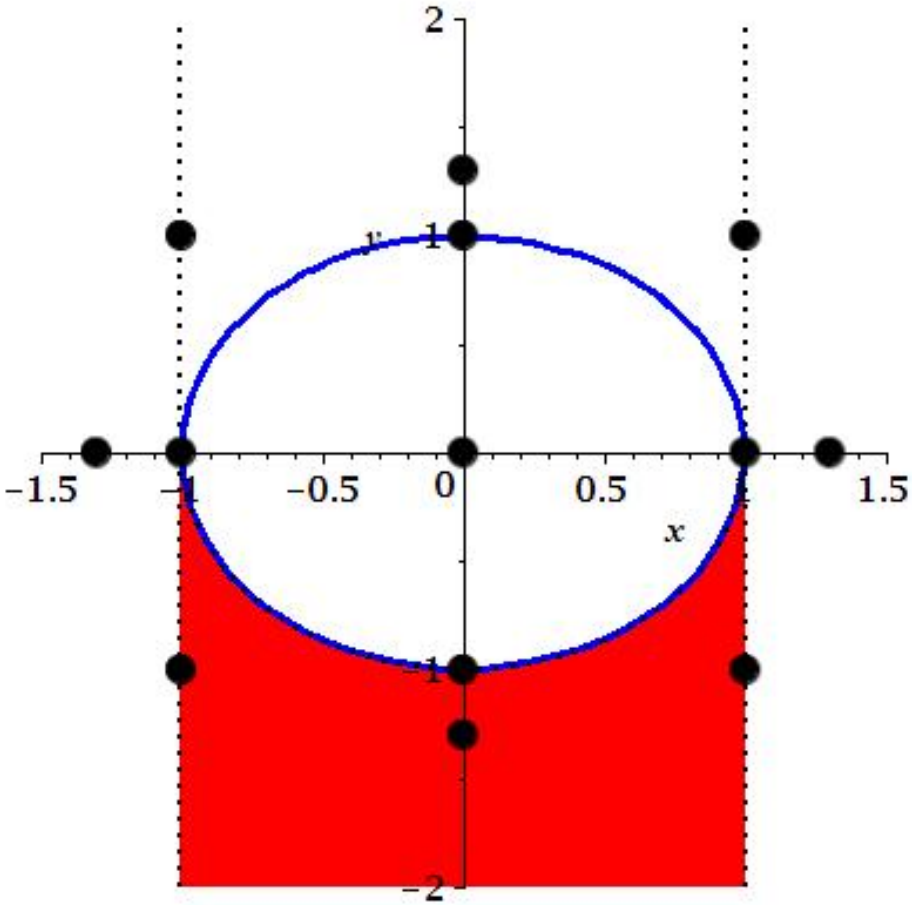


# Sign-invariance



Traditionally a CAD is produced from a set of polynomials such that each polynomial has constant sign (positive, zero or negative) in each cell. Such a CAD is said to be **sign-invariant**.

E.g. A CAD produced to be sign-invariant for the polynomial $f := x^2 + y^2 - 1$ will commonly have 13 cells.

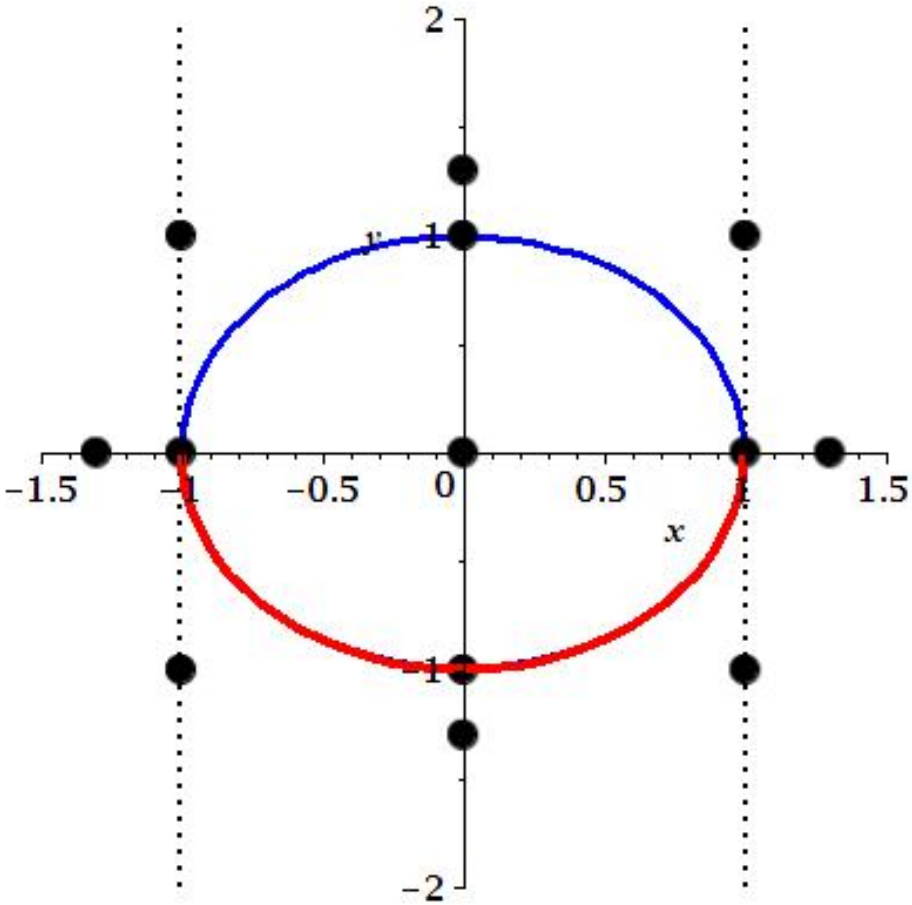


# Sign-invariance



Traditionally a CAD is produced from a set of polynomials such that each polynomial has constant sign (positive, zero or negative) in each cell. Such a CAD is said to be **sign-invariant**.

E.g. A CAD produced to be sign-invariant for the polynomial $f := x^2 + y^2 - 1$ will commonly have 13 cells.

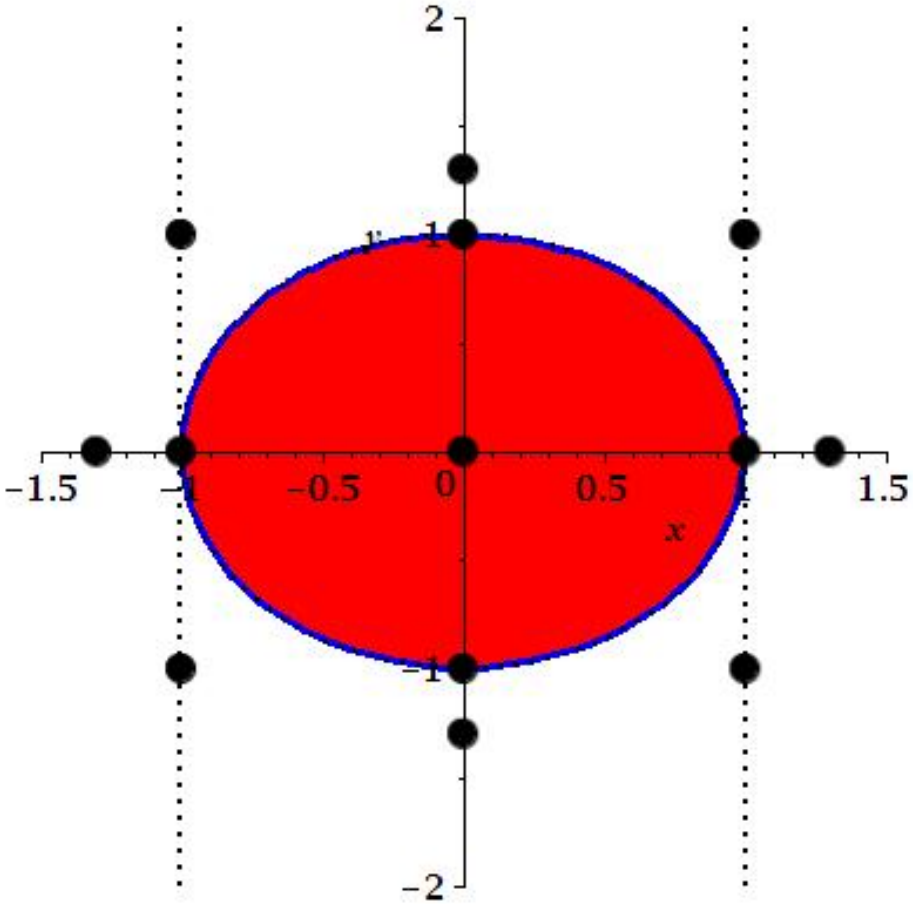


# Sign-invariance



Traditionally a CAD is produced from a set of polynomials such that each polynomial has constant sign (positive, zero or negative) in each cell. Such a CAD is said to be **sign-invariant**.

E.g. A CAD produced to be sign-invariant for the polynomial $f := x^2 + y^2 - 1$ will commonly have 13 cells.

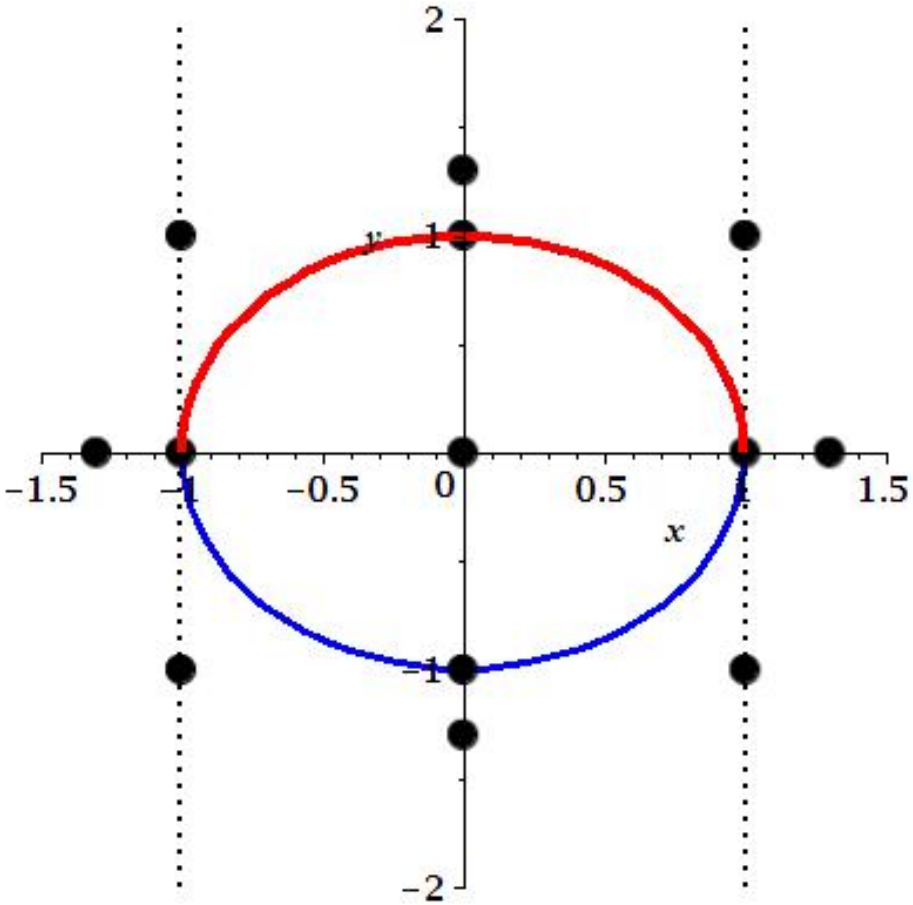


# Sign-invariance



Traditionally a CAD is produced from a set of polynomials such that each polynomial has constant sign (positive, zero or negative) in each cell. Such a CAD is said to be **sign-invariant**.

E.g. A CAD produced to be sign-invariant for the polynomial $f := x^2 + y^2 - 1$ will commonly have 13 cells.

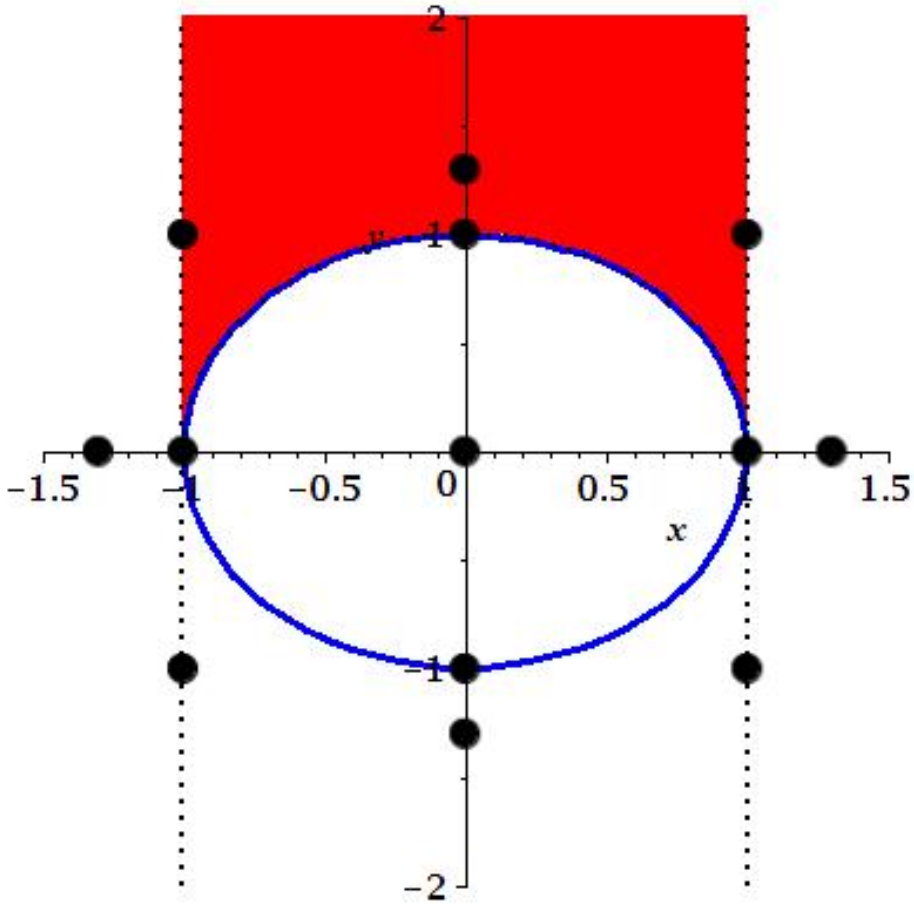


# Sign-invariance



Traditionally a CAD is produced from a set of polynomials such that each polynomial has constant sign (positive, zero or negative) in each cell. Such a CAD is said to be **sign-invariant**.

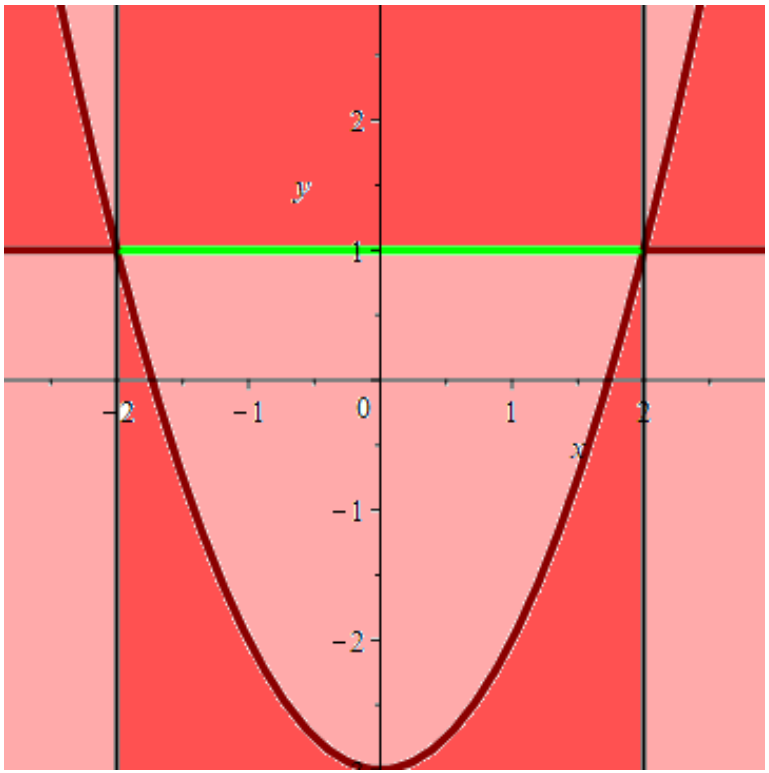
E.g. A CAD produced to be sign-invariant for the polynomial $f := x^2 + y^2 - 1$ will commonly have 13 cells.

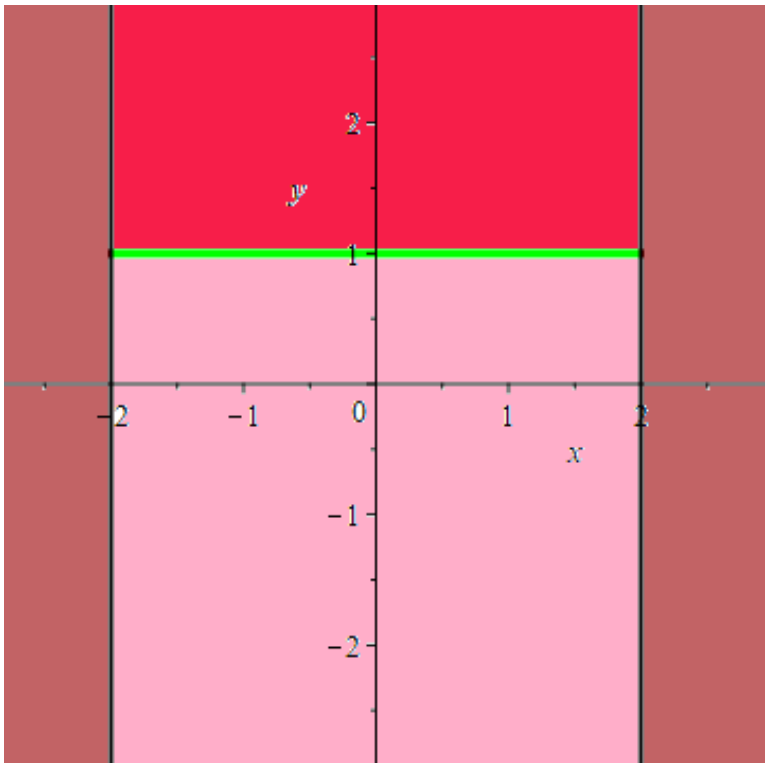


# Truth Invariance



Sign-invariant CAD is more than we need for most applications. We would usually prefer a CAD **truth-invariant** for a formula, which requires fewer cells. E.g. $\phi := x^2 - y < 3 \wedge y = 1$. Left is a sign-invariant CAD and right the coarsest truth-invariant CAD.



# How to achieve truth invariance algorithmically? (Slide 11/28)

There are various approaches to achieve truth-invariance:

- Refine a sign-invariant CAD once produced. However, then no savings in CAD computation itself, only on the future work.
- Terminate lifting if truth can be determined early (a key idea in e.g. Partial CAD).
- Adapt projection according to formula structure.
  I.e. ignore some of the polynomials in some parts of the space when we can conclude they do not affect the truth of the overall formula. This leads us to study "Equational Constraints".

# Equational Constraints



An **Equational Constraint** (EC) is a polynomial equation logically implied by a quantifier-free Tarski formula.

An EC is **explicit** when it is an atom of the input formula, and **implicit** otherwise.

E.g., $f = 0$ is an explicit EC of the formula $\phi = (f = 0) \wedge \varphi$.

E.g., $f_1 f_2 = 0$ is an implicit EC for

$$\psi = (f_1 = 0 \wedge \varphi_1) \vee (f_2 = 0 \wedge \varphi_2).$$

E.g., $f^2 + g^2 \leq 0$ has two implicit ECs: $f = 0$ and $g = 0$.

# Key observation on ECs



In 1998 Collins observed that truth invariance of a formula with equational constraint $f = 0$ is implied by:

- Sign-invariance for $f$; and
- Sign-invariance for all other polynomials $g_i$ when $f = 0$.

He proposed a natural adaptation of the projection stage of the CAD algorithm: but proving the validity of this has been non-trivial.

For example, the projection usually takes resultants of every pair of polynomials (to capture their intersections) but in this case we should need only resultants which involve the EC polynomial.

When the EC is explicit this reduced projection is a subset of the original sign-invariant projection.

# The Impact of ECs on CAD



1998: Basic idea proposed by Collins.

1999: McCallum validates operator for first projection only.

2001: McCallum describes how to propagate and use ECs in subsequent projections.

2015: England-Bradford-Davenport:

- corresponding improvements to lifting stage;
- start to consider effect on complexity.

**ISSAC'15 Ex**
1,118,205 S.I. cells reduced to:

- 11,961, 30,233, 158,475 or 158,451 cells;
- 5633 – 21,079 cells;
- 113, 103 or 93 cells;

depending on EC designation.

# The Impact of ECs on CAD



1998: Basic idea proposed by Collins.

1999: McCallum validates operator for first projection only.

2001: McCallum describes how to propagate and use ECs in subsequent projections.

2015: England-Bradford-Davenport:

- corresponding improvements to lifting stage;
- start to consider effect on complexity.

**ISSAC'15 Ex**
1,118,205 S.I. cells reduced to:

- 11,961, 30,233, 158,475 or 158,451 cells;
- 5633 – 21,079 cells;
- 113, 103 or 93 cells;

depending on EC designation.

# The Impact of ECs on CAD



1998: Basic idea proposed by Collins.

1999: McCallum validates operator for first projection only.

2001: McCallum describes how to propagate and use ECs in subsequent projections.

2015: England-Bradford-Davenport:

- corresponding improvements to lifting stage;
- start to consider effect on complexity.

**ISSAC'15 Ex**
1,118,205 S.I. cells reduced to:

- 11,961, 30,233, 158,475 or 158,451 cells;
- 5633 – 21,079 cells;
- 113, 103 or 93 cells;

depending on EC designation.

# Reduced vs Semi-Reduced EC Projection (Slide 15/28)

Assuming a set of primitive polynomials $B$ of which a subset $F$ forms an EC, we have the following validated CAD projection operators.

McCallum1998 $P(B) = \text{coeff}(B) \cup \text{disc}(B) \cup \text{res}(B)$
Use for order-invariant, and hence sign-invariant, CAD at each projection layer.

McCallum1999 $P_F(B) = P(F) \cup \{\text{res}(f, g) \mid f \in F, g \in B \setminus F\}$
Use for the first projection to achieve a truth-invariant CAD in the case of an EC.

McCallum2001 $P_F^*(B) := P_F(B) \cup \text{disc}(B \setminus F) \cup \text{coeff}(B \setminus F)$
Use for subsequent projections to achieve a truth-invariant CAD in the case of an EC.

Are those extra discriminants for subsequent layers *really* necessary?

# Equational Constraints References


S. McCallum
*An improved projection operator for cylindrical algebraic decomposition.*
Quantifier Elimination and Cylindrical Algebraic Decomposition, pages 242–268, Springer, 1998.


S. McCallum
*On projection in CAD-based quantifier elimination with equational constraints.*
Proc. ISSAC '99, pages 145–149, ACM, 1999.


S. McCallum
*On propagation of equational constraints in CAD-based quantifier elimination.*
Proc. ISSAC '01, pages 223–231, ACM, 2001.


M. England, R. Bradford and J.H. Davenport.
Improving the use of equational constraints in cylindrical algebraic decompositions.
Proc. ISSAC '15, pages 165–172, ACM, 2015


M. England, R. Bradford and J.H. Davenport.
Cylindrical algebraic decomposition with equational constraints
J. Symbolic Computation, 100, pages 38–71, Elsevier, 2020.

# Outline

# Notation



Let $\phi^*$ be a prenex formula of Tarski algebra of the form:

$$(\exists x_{n-k+1}) \dots (\exists x_n) \phi(x_1, \dots, x_n)$$

where $\phi$ is quantifier-free. I.e., $n$ variables with $k$ quantified and $s := n - k$ free. We assume further that

$$\phi = f_1 = 0 \wedge \dots \wedge f_t = 0 \wedge \psi$$

for quantifier-free formula $\psi$, i.e. $f_1, \dots, f_t$ are (explicit) ECs of $\phi^*$.

We assume further an initial segment of the variable sequence is partitioned into:

- **Parameters ($p$):** $x_1, \dots, x_p$, where $p \leq s$; and
- **Unknowns of interest ($u$):** $x_{p+1}, \dots, x_{p+u}$ , where $p + u \leq n$.

# Our Goals



## Goal 1

**Express unknowns of interest in terms of the parameters.**

- Construct a cell decomposition $\mathcal{D}_p$ of the parameter space $\mathbb{R}^p$.
- On each open cell of $\mathcal{D}_p$, the number of associated solutions is either a finite constant or infinite.
- On each such cell with a finite positive constant number of associated solutions, we want an expression for the solutions in terms of the parameters.

## Goal 2

**Achieve efficiency gains where possible.** Move closer to validation of Collins' original fully-reduced CAD equational projection operation.

# Related Work on the first goal



Our first goal is an extension to the scope of Lazard and Rouillier, where the given parametric polynomial systems are assumed to be quantifier-free and to satisfy a number of other restrictions.

Lazard, D., Rouillier, F.

*Solving parametric polynomial systems*.

J. Symbolic Computation 42(6), 636–667, 2007.

https://doi.org/10.1016/j.jsc.2007.01.007

There are also potential connections with the work of Brown on restricted order presented at SYNASC 2023, and the work on border polynomials by Moreno Maza et al.

Moreno Maza, M., Xia, B., Xiao, R.

*On solving parametric polynomial systems*.

Mathematics in Computer Science 6(4), 457–473, 2012.

https://doi.org/10.1007/s11786-012-0136-3

# Our Progress to Date



Goal 1: We have treated the case $p = s$ and $p + u = n$ in detail.

Goal 2: Provided certain conditions are met, we show that the second projection step can use the fully-reduced projection operator (previous theory allowed only a semi-reduced operator).

Full details of the above are available in the arXiv preprint:
`https://doi.org/10.48550/arXiv.2604.23873`
In both cases we think there is scope to go further.

# Outline

# Key Proposition



## Proposition 2

Let $\mathcal{D}$ be a truth-invariant CAD of $\mathbb{R}^n$ for $\phi$ and let $\mathcal{D}_s$ denote the CAD of $\mathbb{R}^s$ induced by $\mathcal{D}$. Let $c$ be a cell of $\mathcal{D}_s$, and let $\alpha \in c$.

Then the total number $\nu_{c,\alpha}$ of distinct solution vectors $\beta \in \mathbb{R}^k$ for $\phi$ over $\alpha$ is an element of $\mathbb{N} \cup \{\infty\}$, constant as $\alpha$ varies in $c$: hence we may define $\nu_c$, the cardinality of the set of solution vectors associated with the points of $c$, to be this number.

Moreover, in the case that $0 < \nu_c < \infty$, the $\nu_c$ distinct solution vectors for $c$ are expressible as continuous functions of the parameters in $c$.

# Algorithm 1



Proposition 2 justifies Goal 1 as achievable, and suggests an approach to achieve it:

- Perform CAD projection with equational constraints.
- Compute an Open CAD of parameter space (i.e. only cells full-dimensional in the parameter space).
- For each cell, test a sample point to find $\nu_c$, the number of distinct solution vectors.
- For each cell where $0 < \nu_c < \infty$, construct the solution function expressions by generalising from the sample point.

This is presented formally in Algorithm 1 of the paper.

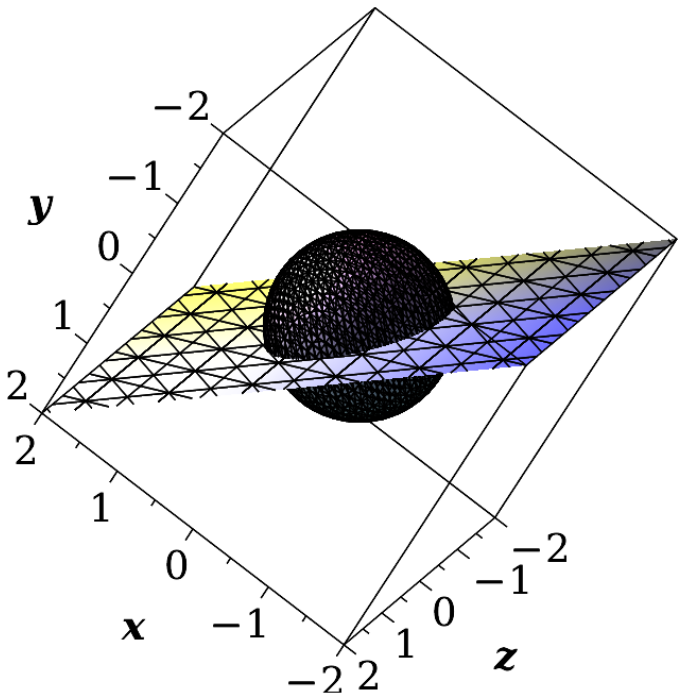
# Simple Example



Let $n = 3$ and $\phi^* \equiv (\exists y)(\exists z)[z^2 + y^2 + x^2 - 1 = 0 \wedge z - y = 0]$. Then parameter space is the $x$-axis and CAD projection followed by Open CAD construction over $\mathbb{R}^1$ will decompose this into three open cells and their sample points as below.

$$c_1 = (-\infty, -1), \quad s_1 = -2$$
$$c_2 = (-1, 1), \quad s_2 = 0$$
$$c_3 = (1, \infty), \quad s_3 = 2$$

At $s_1$ and $s_3$ there are no solutions to $\phi$, while at $s_2$ there exist two: $y = z = \pm 1/\sqrt{2}$. Over $c_2$ we generalise these to: $y = z = \pm\sqrt{(1 - x^2)/2}$.



# Relative Order



The CAD projections reviewed earlier relied on the theory of **order (of vanishing) invariance**, requiring those "extra discriminants".

We now introduce a new concept of the **relative order**: the order of a real analytic function relative to the real variety of another. It captures the extent to which a polynomial vanishes on the variety.

## Definition

Let $p$ be a non-singular point of the variety $V_f$ (i.e. where $f(x) = 0$). Then near $p$, $V_f$ has a local graph:

$$x_i = \phi(x_1, \ldots, \hat{x}_i, \ldots, x_n)$$

where $\hat{x}_i$ denotes omission of variable $x_i$. The **relative order** of $g$ to $V_f$ at $p$ is the order of the analytic function:

$$g(x_1, \ldots, \phi(x_1, \ldots, \hat{x}_i, \ldots, x_n), \ldots, x_n)$$

at $(p_1, \ldots, \hat{p}_i, \ldots, p_n)$. (This is well defined, i.e. independent of $i$.)

# More Efficient CAD



We reformulate the traditional theorems that underpins CAD lifting with ECs using this new concept of relative order.

This allows us to validate the use of Collins' fully-reduced CAD projection for equational constraints in the second projection step under some conditions.

There are more conditions that we would like (or think necessary). Nevertheless, there are practical examples of interest which meet these conditions and benefit from the new theory.

# Validation Conditions



## Validation Conditions

1. Let $f$ be the EC at level $n$ and $e$ the EC at level $n-1$. Then $f$ and $e$ must be **well-placed**: meaning either discrim($f$) is a product of elements of $E$ (the factors of $e$); or $e$ and discrim($f$) are coprime.
2. The cells we lift over in $\mathbb{R}^{n-2}$ must have codimension $\leq 1$.
3. The real variety $V_e$ must be non-singular.

**Conditions 1 and 3 are reasonable:** they express that the equational constraint is really restricting the solution space, and thus that we should expect to benefit from less computation.
**Condition 2 is unreasonable:** it is present because required by a technical result we rest part of the proof on. We are hopefully that an alternative proof technique could remove it.

# Outline

# Solotareff's approximation problem (Slide 26/28)

This asks one to find the best approximation to a real polynomial $x^n + rx^{n-1}$ (where $r \geq 0$) by a real polynomial of degree at most $n-2$. There are explicit theoretical solutions when $0 \leq r \leq S_n = n(\tan(\pi/(2n)))^2$.

Collins showed how for $r > S_n$ the question may be formulated as a QE problem.

Collins, G.E.

*Application of quantifier elimination to Solotareff's approximation problem.*

In: Jeltsch, R., Mansour, M. (eds.) Stability Theory. International Series of Numerical Mathematics, vol. 121, pp. 181–190. Birkhäuser Basel, 1996.

https://doi.org/10.1007/978-3-0348-9208-7_19

# QE Formulation of Solotareff



For $n = 3$:

$$\phi^* \equiv (\exists u)[r > 1 \wedge -1 < u \wedge u < 1 \wedge 3u^2 + 2ru - 1 = 0 \\ \wedge u^3 + ru^2 - u + r - 2b = 0].$$

For $n = 4$:

$$(\exists b)(\exists u)(\exists v)[[r^2 - 24r + 16 < 0 \vee [r > 1 \wedge r^2 - 24r + 16 \geq 0]] \\ \wedge r - b > 0 \wedge -1 < u \wedge u < v \wedge v < 1 \\ \wedge v^4 + rv^3 - av^2 - bv - (1 - a) = r - b \\ \wedge u^4 + ru^3 - au^2 - bu - (1 - a) = r - b \\ \wedge 4v^3 + 3rv^2 - 2av - b = 0 \\ \wedge 4u^3 + 3ru^2 - 2au - b = 0].$$

# Solotareff Solutions Validated



The case $n = 2$ is easy to solve by hand. Collins (1996) gave a computer-assisted solution for $n = 3,\ 4$. However, his CAD computations made use of the not-yet-validated reduced projection.

In 1999, McCallum validated the use of the reduced operator for the first projection, and thus the $n = 3$ solution.

McCallum, S.

*On projection in CAD-based quantifier elimination with equational constraint.*

In: Proceedings of ISSAC '99. pp. 145–149. ACM Press, 1999.

https://doi.org/10.1145/309831.309892

We show that the validation we give for the reduced second projection applies to this problem, and thus validate Collins' solution in the $n = 4$ case, 30 years after it was first written.

# Thanks for Listening!