# Automated Root Certification in Lean

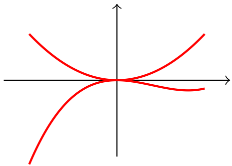
## for Transversal Polynomial Systems

Dattabhasvant Ghadiyaram

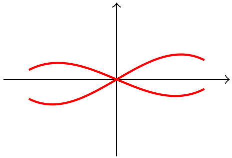
Indian Instititute of Science, Bengaluru

# Introduction

- Given a box in $\mathbb{R}^n$ and a transversal system of polynomial equations, search for roots and certify their existence or absence.
- Transversality: Jacobian is locally invertible.
- Automated the root search and the construction of the corresponding formal proof term.



Tangential



Transversal

# Motivation

$$G = \left\langle a, b \mid aba^{-1} = b^2,\ bab^{-1} = a^2 \right\rangle$$

# Motivation

$$G = \langle a, b \mid aba^{-1} = b^2,\ bab^{-1} = a^2 \rangle$$

- This group is trivial, i.e. G has only the identity element
- Triviality of group presentations is undecidable, i.e. there is no general algorithm.

# Proving Nontriviality



- However, we can prove non-triviality if there is a non-trivial representation in a known group.

# Proving Nontriviality



- However, we can prove non-triviality if there is a non-trivial representation in a known group.

$$H = \langle x, y \mid xyx = yxy \rangle$$

# Proving Nontriviality



- However, we can prove non-triviality if there is a non-trivial representation in a known group.

$$H = \langle x, y \mid xyx = yxy \rangle$$

- Consider $\phi : H \to SU(2)$, where $\phi(x) = X$ and $\phi(y) = Y$:

$$X = \begin{pmatrix} x_1 + ix_2 & 0 \\ 0 & x_1 - ix_2 \end{pmatrix} \qquad Y = \begin{pmatrix} y_1 + iy_2 & y_3 \\ y_3 & y_1 - iy_2 \end{pmatrix}$$

# Proving Nontriviality



- However, we can prove non-triviality if there is a non-trivial representation in a known group.

$$H = \langle x, y \mid xyx = yxy \rangle$$

- Consider $\phi : H \to SU(2)$, where $\phi(x) = X$ and $\phi(y) = Y$:

$$X = \begin{pmatrix} x_1 + ix_2 & 0 \\ 0 & x_1 - ix_2 \end{pmatrix} \qquad Y = \begin{pmatrix} y_1 + iy_2 & y_3 \\ y_3 & y_1 - iy_2 \end{pmatrix}$$

- We need non-identity matrices $X, Y \in SU(2)$ which satisfy:

$$\det X = 1 \qquad \det Y = 1 \qquad XYX = YXY$$

# Polynomial System



Reduces to solving a system of 5 polynomial equations in 5 variables:

$$
\begin{aligned}
x_1^2 + x_2^2 &= 1 \\
y_1^2 + y_2^2 - y_3^2 &= 1 \\
(x_1^2 - x_2^2)y_1 - 2x_1x_2y_2 - x_1y_1^2 + x_1y_2^2 + 2x_2y_1y_2 - x_1y_3^2 &= 0 \\
(x_1^2 - x_2^2)y_2 + 2x_1x_2y_1 - 2x_1y_1y_2 - x_2y_1^2 + x_2y_2^2 + x_2y_3^2 &= 0 \\
y_3(x_1^2 + x_2^2 - 2x_1y_1 + 2x_2y_2) &= 0
\end{aligned}
$$

# Symbolic Methods



- Classical symbolic tools include Gröbner bases and cylindrical algebraic decomposition.
- In the worst case, these methods have doubly exponential complexity in the number of unknowns.
- For our use case, this grows quickly: more generators, or larger target groups such as $SU(n)$ and $SO(n)$, introduce many new variables.

# Numerical Methods



- Floating Point Approximation: $\pi \to 3.14159$
- When we evaluate some $f$, we only know $f(3.14159) \approx f(\pi)$
- For proofs, separately need to reason $|f(\pi) - f(3.14159)| < \epsilon$ because $|\pi - 3.14159| < \delta$

# Interval Arithmetic



- $\pi \to [3215/1024, 3217/1024]$ (Computable Endpoints)
- We chose dyadic rationals, i.e. $p/2^k$ for an odd $p \in \mathbb{N}$ and $k \in \mathbb{Z}$
- Can we apply $f$ to $[\frac{3215}{1024}, \frac{3217}{1024}]$ and obtain $dy[[a, b]]$ containing the range of $f$?

# Arithmetic Operations

- **Addition** : `dy[[a,b]] + dy[[c,d]] = dy[[a+c, b+d]]`
- **Subtraction** : `dy[[a,b]] - dy[[c,d]] = dy[[a-d, b-c]]`

# Arithmetic Operations



- **Addition** : `dy[[a,b]] + dy[[c,d]] = dy[[a+c, b+d]]`
- **Subtraction** : `dy[[a,b]] - dy[[c,d]] = dy[[a-d, b-c]]`
- **Multiplication** : `dy[[a,b]] * dy[[c,d]] = dy[[min(ac, ad, bc, bd), max(ac, ad, bc, bd)]]`
- **Division (Approximate)** : If $0 \notin [[c, d]]$,

$$\frac{[[a, b]]}{[[c, d]]} = [[\min(a/c, a/d, b/c, b/d), \max(a/c, a/d, b/c, b/d)]]$$

rounded outwards to the nearest dyadic of specified precision.

# Arithmetic Operations



- **Addition** : dy[[a,b]] + dy[[c,d]] = dy[[a+c, b+d]]
- **Subtraction** : dy[[a,b]] - dy[[c,d]] = dy[[a-d, b-c]]
- **Multiplication** : dy[[a,b]] * dy[[c,d]] = dy[[min(ac, ad, bc, bd), max(ac, ad, bc, bd)]]
- **Division (Approximate)** : If $0 \notin [[c, d]]$,

$$\frac{[[a, b]]}{[[c, d]]} = [[\min(a/c, a/d, b/c, b/d), \max(a/c, a/d, b/c, b/d)]]$$

  rounded outwards to the nearest dyadic of specified precision.
- **Exponentiation**
  - dy[[a,b]] $^{2k+1}$ = dy[[a$^{2k+1}$, b$^{2k+1}$]]
  - dy[[-a,b]] $^{2k}$ = dy[[0, max((-a)$^{2k}$, b$^{2k}$)]]
  - dy[[a,b]] $^{2k}$ = dy[[min(a$^{2k}$, b$^{2k}$), max(a$^{2k}$, b$^{2k}$)]]

# Properties

- **Soundness**: if $I \cdot J = K$, then every actual product $xy$ with $x \in I$ and $y \in J$ lies inside $K$.
- **Sharpness**: for addition, subtraction, multiplication, and powers, every point of the output interval is actually attained by some valid input values.
- Division is sound, but outward rounding means it need not be sharp.
- Exponentiation as repeated multiplication is not sharp!

# Dependency Problem

- No additive inverse : $I - I \neq 0$

$$dy[[-1,1]] - dy[[-1,1]] = dy[[-1-1, 1-(-1)]] = dy[[-2,2]]$$

# Dependency Problem

- No additive inverse : $I - I \neq 0$

$$dy[[-1,1]] - dy[[-1,1]] = dy[[-1-1, 1-(-1)]] = dy[[-2,2]]$$

- Distributivity fails : $I * (J + K) \neq I * J + I * K$

$$dy[[-1,2]] * (dy[[1,2]] + dy[[-2,-1]]) = dy[[-2,2]]$$
$$dy[[-1,2]] * dy[[1,2]] + dy[[-1,2]] * dy[[-2,-1]] = dy[[-6,6]]$$

# Dependency Problem



- No additive inverse : $I - I \neq 0$

$$dy[[-1,1]] - dy[[-1,1]] = dy[[-1-1, 1-(-1)]] = dy[[-2,2]]$$

- Distributivity fails : $I * (J + K) \neq I * J + I * K$

$$dy[[-1,2]] * (dy[[1,2]] + dy[[-2,-1]]) = dy[[-2,2]]$$
$$dy[[-1,2]] * dy[[1,2]] + dy[[-1,2]] * dy[[-2,-1]] = dy[[-6,6]]$$

- Overestimation when evaluating polynomials.
  Consider $f(x) = x^2 - x$ over dy[[0,1]]

True Range :$dy[[-0.25, 0]]$ Computed Range :$dy[[-1, 1]]$

# Vectors and Matrices

- **Interval Vector**: a box in $\mathbb{R}^n$

$$\mathbf{X} = ([a_1, b_1], [a_2, b_2], \ldots, [a_n, b_n])$$

- **Interval Matrix**: a matrix whose entries are intervals

$$A = \begin{pmatrix} [a_{11}, b_{11}] & \cdots & [a_{1n}, b_{1n}] \\ \vdots & \ddots & \vdots \\ [a_{m1}, b_{m1}] & \cdots & [a_{mn}, b_{mn}] \end{pmatrix}$$

- Standard operations : Addition, Subtraction, Dot Product, Matrix-Vector multiplication, matrix multiplication

# Interval Extensions



- For a function $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$, an **Interval Extension** is any interval valued function $F$, such that $\forall \mathbf{x} \in \mathbb{R}^n, F(\mathbf{x}) = f(\mathbf{x})$
- Inclusion Isotonicity : $\mathbf{X} \subseteq \mathbf{Y} \Rightarrow F(\mathbf{X}) \subseteq F(\mathbf{Y})$

# Interval Extensions

- For a function $f : \mathbb{R}^n \to \mathbb{R}^m$, an **Interval Extension** is any interval valued function $F$, such that $\forall \mathbf{x} \in \mathbb{R}^n, F(\mathbf{x}) = f(\mathbf{x})$
- Inclusion Isotonicity : $\mathbf{X} \subseteq \mathbf{Y} \Rightarrow F(\mathbf{X}) \subseteq F(\mathbf{Y})$
- For well-behaved functions, **Mean Value Form**:

$$F_{mv}(\mathbf{X}) = f(\mathbf{x_m}) + \nabla F((\mathbf{X})) \cdot (\mathbf{X} - \mathbf{x_m})$$
$$= f(\mathbf{x_m}) + \sum_{i=1}^{n} D_i F(\mathbf{X}) \cdot (X_i - x_{m,i})$$

- By the Mean Value theorem for $f$, $\forall \mathbf{x} \in X, f(\mathbf{x}) \in F_{mv}(X)$

# Computable Polynomials



- `p : MvRatPol n` represents $p \in \mathbb{Q}[X_1, \ldots, X_n]$; strictly a list of $\mathbb{Q} \times \mathbb{N}^n$ tuples
- Can be evaluated on $\mathbf{x} \in \mathbb{R}^n$ for proofs and on $\mathbf{X}$, an interval vector for computation
- Symbolic derivative `MvRatPol n` $\rightarrow$ `MvRatPol n`
- `System m n` represents a system with $m$ polynomial equations in $n$ unknown; precisely $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ where each component is a polynomial described above

# Interval Newton Method

- Newton-Raphson Method :

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

- Interval Newton:

$$N(I) = I_m - \frac{f(I_m)}{F'(I)} \qquad \text{if } 0 \notin F'(I)$$

- $N(I) \subseteq I \Rightarrow I$ contains exactly one root of $f$
- $N(I)$ and $I$ Disjoint $\Rightarrow I$ has no root of $f$.
- MV Newton-Raphson:

$$\mathbf{x_{n+1}} = \mathbf{x_n} - J_S(\mathbf{x_n})^{-1} S(\mathbf{x_n})$$

# Krawczyk Method



- For a pre-conditioner rational matrix $Y$, define the interval-vector valued Krawczyk operator;

$$K(\mathbf{X}) = \mathbf{x_m} - Y \cdot S(\mathbf{x_m}) + (I - Y * J_S(\mathbf{X})) \cdot (\mathbf{X} - \mathbf{x_m})$$

- $Y$ is chosen as inverse of the midpoint-matrix of $J_S(X)$

**Existence & Uniqueness**

If $\|I - YJ_S(\mathbf{X})\| < 1$ and $\det(Y) \neq 0$
$K(\mathbf{X}) \subseteq \mathbf{X} \Rightarrow \mathbf{X}$ contains exactly one root of $S$

**Exclusion**

$K(\mathbf{X}) \cap \mathbf{X} = \emptyset \Rightarrow \mathbf{X}$ has no root of $S$

# Proof of Correctness



Define the pointwise Krawczyk function $k : \mathbb{R}^n \to \mathbb{R}^n$ given by $k(\mathbf{x}) = \mathbf{x} - Y \cdot S(\mathbf{x})$.

**Soundness**

$\forall \mathbf{x} \in \mathbf{X},\ \ k(\mathbf{x}) \in K(\mathbf{X})$, equivalently $k(\mathbf{X}) \subseteq K(\mathbf{X})$

# Proof of Correctness

Define the pointwise Krawczyk function $k : \mathbb{R}^n \to \mathbb{R}^n$ given by $k(\mathbf{x}) = \mathbf{x} - Y \cdot S(\mathbf{x})$.

## Soundness

$\forall \mathbf{x} \in \mathbf{X},\ k(\mathbf{x}) \in K(\mathbf{X})$, equivalently $k(\mathbf{X}) \subseteq K(\mathbf{X})$

- Hypotheses

$$K(\mathbf{X}) \subseteq \mathbf{X} \qquad \|I - YJ_S(\mathbf{X})\| < 1 \qquad \det(Y) \neq 0$$

- $\|k'(\mathbf{x})\| = \|I - YJ_S(\mathbf{x})\| \leq \|I - YJ_S(\mathbf{X})\| < 1$
  $\Rightarrow$ $k$ is a contracting map on $\mathbf{X}$
- BFPT $\Rightarrow \exists! \mathbf{x} \in X$ such that $k(\mathbf{x}) = \mathbf{x} \Rightarrow Y \cdot S(\mathbf{x}) = 0$
- $\det(Y) \neq 0 \Rightarrow S(\mathbf{x}) = 0$

# Algorithm

**Algorithm 1** IsolateRoots

**Input :** $S \leftarrow$ `System n n`, $X \leftarrow$ `Vecterval n`, maxDepth, precision
**Output :** $(C, U)$
**if** $\mathbf{0} \notin S(X)$ **then return** $(\emptyset, \emptyset)$ ▷ Pruned
**if** $det(Y) \neq 0$ and $\|I - YJ_S(X)\| < 1$ **then**
    Compute $K(X)$
    **if** $X \cap K(X) = \emptyset$ **then return** $(\emptyset, \emptyset)$ ▷ Pruned
    **if** $K(X) \subseteq X$ **then return** $(\{X\}, \emptyset)$ ▷ Certified
**end if**
**if** maxDepth $= 0$ **then return** $(\emptyset, \{X\})$ ▷ Unresolved
$(X_1, X_2) \leftarrow$ Split(X) ▷ Bisect
$(C_1, U_1) \leftarrow$ IsolateRoots(S, $X_1$, maxDepth - 1)
$(C_2, U_2) \leftarrow$IsolateRoots(S, $X_2$, maxDepth - 1)
**return** $(C_1 \cup C_2, U_1 \cup U_2)$

# Automation

- Our tactic: `krawcheck`
- The tactic grid-searches for suitable parameters to run the algorithm computational precision and maximum recursion depth.
- It runs `IsolateRoots`; if the result matches the target statement, it closes the goal by constructing the Lean proof term.

```
def V₀ : Vecterval 2 := #v[dy[[0,1/2]], dy[[0,1]]]
def V₁ : Vecterval 2 := #v[dy[[-1,0]], dy[[0,1]]]
def V₂ : Vecterval 2 := #v[dy[[3/2, 2]], dy[[-5/2, 2]]]
def S : System 2 2 := poly[x1^3, x2^3, -3/2 * x1 * x2;
  4 * x1^2 * x2, 9/4 * x1 * x2^2, -1/2 * x1, -5/2 * x2, 1]

example : V₀.HasRoot S := by krawcheck
example : V₁.HasNoRoot S := by krawcheck
example : V₂.HasUniqueRoot S := by krawcheck
```

# Future Work

- Build on this framework to formalize certification of non-triviality for group presentations.
- Couple the tactic with an AI agent that guesses target groups and candidate representations, which the tactic can then verify.
- Improve the search algorithm to reduce runtime and scale to larger polynomial systems.