Solving polynomial systems over the reals exactly: why and how?

Mohab Safey El Din
UPMC/INRIA/CNRS

2015
Real solution set $S \subset \mathbb{R}^n$ to

$$F_1 = \cdots = F_p = 0, \quad G_1 > 0 \ldots, G_s > 0$$

with $F_i$ and $G_j$ in $\mathbb{Q}[X_1, \ldots, X_n]$

Algorithmic specifications.

- Decide the non-emptiness of $S$ and compute sample points in $S$;
- Answer connectivity queries on $S$;
- “Compute” the projection of $S$ on some given linear space

$$\exists X \in \mathbb{R} \quad X^2 + bX + c = 0 \iff b^2 - 4c \geq 0$$
Symbolic/Exact computation

- General software (among many others)

![Sage](sage.png) ![Maple](maple.png) ![Mathematica](mathematica.png)

among many others...
Symbolic/Exact computation

- General software (among many others)

- Symbolic computation / computer algebra.
  - Efficient tools for basic algebraic/exact computations
  - Large algorithmic scope (arithmetic → differential algebra)
  - BUT **algebraic** manipulations on polynomial expressions give information on **complex roots**

among many others...
Symbolic/Exact computation

- General software (among many others)

- Symbolic computation / computer algebra.
  - Efficient tools for basic algebraic/exact computations
  - Large algorithmic scope (arithmetics → differential algebra)
  - BUT algebraic manipulations on polynomial expressions give information on complex roots

- Reliability issues – crucial for decision problems/algebraic nature

- Efficiency issues – complexity may be exponential in $n$
Symbolic/Exact computation

- General software (among many others)
  - Symbolic computation / computer algebra.
    - Efficient tools for basic algebraic/exact computations
    - Large algorithmic scope (arithmetics → differential algebra)
    - BUT algebraic manipulations on polynomial expressions give information on complex roots
  - Reliability issues – crucial for decision problems/algebraic nature
  - Efficiency issues – complexity may be exponential in $n$

Goal of this talk

Show how to use specialized software tools to solve “hard” problems
Real root classification problem

The dream team

Two View Geometry

Epipolar plane

\( \pi \)

\( X, x, y, C, C_0 \) all coplanar

\( \text{rank}(F) = 2 \) s.t.

Epipolar equation

\( a_3 \cdot F \cdot a_3^\top = 0 \)

Fundamental matrix

7 degrees of freedom

If cameras are calibrated get essential matrix

5 degrees of freedom

\[ F_1(X, Y) = \cdots = F_p(X, Y) = 0 \]

with \( X = (X_1, \ldots, X_n) \) and \( Y = (Y_1, \ldots, Y_t) \)

- Generic finite number of complex roots w.r.t \( Y \)
- To have more fun, we also may consider

\[ G_1(X, Y) > 0, \ldots, G_s(X, Y) > 0 \]
How to solve real root classification problems?

1. Compute a set containing the boundary

   - Projection
   - Elimination

   Gröbner bases

Many other tools are available: geometric resolution, triangular sets, resultants, etc.
How to solve real root classification problems?

1. Compute a set containing the boundary
   - Projection
   - Elimination
   - Gröbner bases

2. Compute sample points outside the boundary
   - Polynomial Optimization

Many other tools are available: geometric resolution, triangular sets, resultants, etc.
How to solve real root classification problems?

1. Compute a set containing the boundary

   - Projection
   - Elimination

   Gröbner bases

2. Compute sample points outside the boundary

   Polynomial Optimization

3. Lift and count the number of solutions

   Gröbner bases

Many other tools are available: geometric resolution, triangular sets, resultants, etc.
First projection step

- Need to characterize tangent spaces
- They are kernels of $\text{jac}(F)$ under regularity assumptions
- Elimination of the $X$-variables in the system $F, \text{MaxMinors}(\text{jac}(F, X))$.
- Inequalities handled by considering $\text{Sols}(F) \cap \text{Sols}(G_i)$.
The big polynomial degree 12, 14 variables. 4251 monomials. BUT degree 4 in each variable other very special property That's a bingo!! Tarantino, TIB
The big polynomial

- degree 12, 14 variables.
- 4251 monomials.
The big polynomial

- degree 12, 14 variables.
- 4251 monomials.

BUT

- degree 4 in each variable
- other very special property

That’s a bingo!!

Tarantino, TIB
Second step: sampling
Second step: sampling

\[ F = 0, \ G > 0 \]

Intuitive ideas underlying the algorithm sufficient to find polynomials "capturing" the boundary \( B \) of the solution set.

Fact: for all \( y \in B \), there exists \( x \in \mathbb{R}^n \) s.t.

\[ G(x) = f(x, y) = 0. \]

\[ \lim \stackrel{\not=}{\to} 0 \quad \text{crit}(\varphi, V(G,f^\prime)) \]

\[ V(h_G, 1_i : h_G, i_1 + h_f i) \]
Second step: sampling

\[ F = 0, \ G = \varepsilon \]
Second step: sampling

\[ F = 0, \ G = \varepsilon \]
Second step: sampling

\[ F = 0, \ G = \varepsilon \]
Second step: sampling

\[ F = 0, \quad G = \epsilon \]
Second step: sampling

\[ F = 0, \ G = \varepsilon \]
Second step: sampling

\[ F = 0, \ G = \varepsilon \]
Second step: sampling

\[ F = 0, \ G = \varepsilon \]
Second step: sampling

\[ F = 0, \ G = \varepsilon \]
Second step: sampling

\[ F = 0, \ G = \varepsilon \]
Second step: sampling

\[ F = 0, \ G = \varepsilon \]
Second step: sampling

\[ F = 0, \, G = \varepsilon \]
Second step: sampling

\[ F = 0, \ G = \varepsilon \]
Second step: sampling

\[ F = 0, \ G = \varepsilon \]
Second step: sampling

\[ F = 0, \ G = \varepsilon \]
The big polynomial

- Many critical point loci are actually empty (!)
- All computations done within 12 hours.
- 504 points

\[ \sim \text{number of chambers} \leq 504 \]

That’s a SUPER bingo !!
2-nd and 3-rd steps: zero-dimensional systems

System of equations

Gröbner basis

\[ q(T) = 0, \quad X_1 = q_1(T), \ldots, X_n = q_n(T) \]

State-of-the-art algorithms and implementation

FGB (Faugère) \( \sim \approx 10 \,000 \) complex solutions.
2-nd and 3-rd steps: zero-dimensional systems

- Variety of computational tools allow to solve large problems exactly (sometimes)

- Particularly useful for decision problems or computing topological informations

- “Fast” computational tools: FGB (Faugère) → Gröbner bases
  RAGlib: library for real algebraic geometry (built on top of FGB).
Variety of computational tools allow to solve large problems exactly (sometimes)

Particularly useful for decision problems or computing topological informations

“Fast” computational tools: FGB (Faugère) \(\rightarrow\) Gröbner bases

RAGlib: library for real algebraic geometry (built on top of FGB).

Special algorithms for special structured problems (e.g. LMI’s)