Real roots of algebraic systems of equations and inequations

Fabrice Rouillier

SPACES project - INRIA / LIP6

Presentation

Real roots of algebraic systems of equations and inequations.

K is an ordered field (in practice Q), C its algebraic closure, R its real closure.

The system of equations/inequations to be solved :

- Equalities

$$V = \{f_1(X, p) = 0, \ldots f_s(X, p) = 0\}$$

- Inequalities

$$U = \{h_1(X, p) > 0, \ldots h_l(X, p) > 0\}$$

where $f_i, h_i \in K[X, p] = K[X_1, \ldots, X_n, p_1, \ldots, p_d]$

Presentation

Real roots of algebraic systems of equations and inequations.

K is an ordered field (in practice Q), C its algebraic closure, R its real closure.

The system of equations/inequations to be solved :

- Equalities

$$V = \{f_1(X, p) = 0, \ldots f_s(X, p) = 0\}$$

- Inequalities

$$U = \{h_1(X, p) > 0, \ldots h_l(X, p) > 0\}$$

where $f_i, h_i \in K[X, p] = K[X_1, \ldots, X_n, p_1, \ldots, p_d]$

Study $V \bigcap U \bigcap (R^n \times R^d)$

Goals

Real roots of algebraic systems of equations and inequations.

Goals

Real roots of algebraic systems of equations and inequations.

Algorithms :

Goals

Real roots of algebraic systems of equations and inequations.

Algorithms :

- Exact / certified results

Goals

Real roots of algebraic systems of equations and inequations.

Algorithms :

- Exact / certified results
- Implemented algorithms

Goals

Real roots of algebraic systems of equations and inequations.

Algorithms :

- Exact / certified results
- Implemented algorithms
- Checkable restrictions

Goals

Real roots of algebraic systems of equations and inequations.

Algorithms :

- Exact / certified results
- Implemented algorithms
- Checkable restrictions

Exact computations :

Goals

Real roots of algebraic systems of equations and inequations.

Algorithms :

- Exact / certified results
- Implemented algorithms
- Checkable restrictions

Exact computations :

- Global results

Goals

Real roots of algebraic systems of equations and inequations.

Algorithms :

- Exact / certified results
- Implemented algorithms
- Checkable restrictions

Exact computations :

- Global results
- Study of degenerated cases

Goals

Real roots of algebraic systems of equations and inequations.

Algorithms :

- Exact / certified results
- Implemented algorithms
- Checkable restrictions

Exact computations :

- Global results
- Study of degenerated cases
- Time and memory consuming

Goals

Real roots of algebraic systems of equations and inequations.

Algorithms :

- Exact / certified results
- Implemented algorithms
- Checkable restrictions

Exact computations :

- Global results
- Study of degenerated cases
- Time and memory consuming

Certified computations :

Goals

Real roots of algebraic systems of equations and inequations.

Algorithms :

- Exact / certified results
- Implemented algorithms
- Checkable restrictions

Exact computations :

- Global results
- Study of degenerated cases
- Time and memory consuming

Certified computations :

- use of certified semi-numerical technics;

Goals

Real roots of algebraic systems of equations and inequations.

Algorithms :

- Exact / certified results
- Implemented algorithms
- Checkable restrictions

Exact computations :

- Global results
- Study of degenerated cases
- Time and memory consuming

Certified computations :

- use of certified semi-numerical technics;
- mix semi-numerical and exact computations;

Plan

Part 1 : Univariate polynomials with rational coefficients

- Resultants;
- Sturm Sequences;
- Descartes' rule of signs

Part 2 : A Universal Solution - the C.A.D

Part 3 : basic computable objects

- Working with ideals : Gröbner bases;
  - canonical representation;
  - properties (dimension, degree);
- *Describing* complex Solutions :
  - the shape of lexicographic Gröbner bases;
  - triangular sets;
  - Systems with a finite number of complex roots : the R.U.R;

Plan

Part 4 : Real Roots

- Zero dimensional systems
- Univariate Polynomials with real algebraic numbers as coefficients
- Positive Dimensional Algebraic Systems

Examples

- Through 5 examples, several technics for *solving* zero-dimensional systems in practice : how to choose the right strategy ...
- One large example of system with parameters and inequalities;

Polynômes en une variable : nbre de racines complexes distinctes

$$p = \sum_{i=0}^{D} a_i X^i = \prod_{i=0}^{d} (X - \alpha_i)^{\mu_i}$$

C'est d !

Attention au pgcd ! Si $p \in \mathbb{Q}[X]$, $d = degree(p/pgcd(p, p'))$, sinon, attention !

Exemple : $p = aX^2 + bX + c \in \mathbb{Q}[a, b, c, X]$: $pgcd(p, \frac{\partial p}{\partial x}) = 1$, ce qui est tout à fait normal ($p = (x-1) * (x+1)$) mais il est faux en general que $pgcd(p, \frac{\partial p}{\partial x})_{a=a_0,\ldots,c=c_0} = pgcd((p)_{a=a_0,\ldots,c=c_0}, (\frac{\partial p}{\partial x})_{a=a_0,\ldots,c=c_0})$ (exemple $p = (x-1)^2$).

Quelques armes pour travailler avec des coefficients indeterminés ou paramètres sont les résultants et les sous-résultant.

Le Résultant

$F = \sum_{i=0}^{p} a_i X^i$ et $G \sum_{i=0}^{q} b_i X^i$ deux polynômes dans $K[X]$

$$R_X(F,G) = a_p^q b_q^p \prod_{i=1\ldots p, j=1\ldots q} (\alpha_i - \beta_j)$$

Pour le calculer, on peut presque utiliser l'algorithme d'Euclide :
$m = def(F,X)$, $n = deg(G,X)$, $g = lc(G,X)$

- si $n = 0$ alors retourner g^m
- sinon
  - $R = Reste(F,G,X)$; $p = deg(R,X)$;
  - si $R = 0$ retourner 0
  - sinon retourner $(-1)^{np} g^{m-p}$Resultant(G,R);

Le Résultant

- Le résultant de F et G est un élément de K.
- Le resultant de F et G est nul si et deulement si (leurs 2 termes de tête sont nuls ou ils ont un facteur commun non trivial).

Exemple : $F = a * X^2 + b * X + c$ et $G = F' \Rightarrow$ Resultant(F,G)=$a(b^2 - 4 * a * c)$.

Le Résultant

$$R_X(F,G) = det(Syl_X(F,G))$$

$$Syl_X(F,G) = \underbrace{\begin{pmatrix} a_p & \cdots & \cdots & \cdots & \cdots & a_0 & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & a_p & \cdots & \cdots & \cdots & \cdots & a_0 \\ b_q & \cdots & \cdots & \cdots & b_0 & & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & & b_q & \cdots & \cdots & \cdots & b_0 \end{pmatrix}}_{p+q} \begin{matrix} \} q \\ \\ \\ \} p \\ \\ \\ \end{matrix}$$

Majorer le nombre de racines réelles

Notation : $V(a_1) = 0$,

$$Var(a_1, \cdots, a_k) = \begin{cases} Var(a_1, \cdots, a_{k-1}) + 1 & \text{if } \operatorname{sign}(a_{k-1}a_k) = -1 \\ Var(a_1, \cdots, a_{k-1}) & \text{otherwise} \end{cases}$$

Théorème (Descartes) :

- $P = \sum_{i=0}^{d} a_i x^i \in \mathbb{R}[x]$;
- $Var(P) = Var(a_0, \ldots, a_d)$;
- $\operatorname{pos}(P)$ nombre de racines réelles de P strictement positives.

$$\boxed{\operatorname{pos}(P) \leq Var(P) \text{ et } Var(P) - \operatorname{pos}(P) \text{ est pair}}$$

$\operatorname{pos}(P)$ = nombre de racines strictement positives de P comptées avec multiplicités.

Compter les racines réelles

On construit par presque l'algorithme d'euclide, une suite de Sturm associée à F :

- $St_0 = F$, $St_1 = F'$,
- $St_2 = -Reste(St_1, St_2)$,..., $St_i = -Reste(St_{i-1}, St_{i-2})$,...,
- $St_{s+1} = 0$.

Théorème (Sturm) :

- $Var(St_1(b), \ldots, St_s(b)) - Var(St_1(a), \ldots, St_s(a))$ est exactement égal au nombre de racines distinctes de F dans $]a, b[$.

Remarque :
$Var(St_1(+\infty), \ldots, St_s(+\infty)) - Var(St_1(-\infty), \ldots, St_s(-\infty))$ est exactement égal au nombre de racines distinctes de F.

Remarque : Comme le pgcd, la suite de Sturm ne se spécialise pas.

Exemple

$F = x^2 - 1$

Une suite de Sturm de F est $[x^2 - 1, 2x, 1]$

Variations en $-\infty$ = 2

Variations en $+\infty$ = 0

Variations en 0 = 1

On en déduit : 2 racines réelles dans $]-\infty; +\infty[$, 1 racine réelle dans $]-\infty; 0[$ et une racine réelle dans $]0; +\infty[$.

Isoler les zéros réels, un algorithme simple

$P = \sum_{i=0}^{d} a_i X^i \in \mathbb{Q}[X]$ sans facteurs carrés. Kronecker :

- Calcul d'une borne sup M sur le module des racines;
- Calcul d'une borne inf $sep(P)$ sur la distance entre les racines;
- Découpage de $[-M; M]$ en intervalles de largeur au plus $sep(P)$;
- Evaluation de P aux bornes de ces intervalles

Rappel :

- $M < 2max_{i=0}^{d-1}|a_i|^{1/(d-i)}$ (bon ok il y a mieux)
- $sep(P) > \sqrt{\frac{3}{d^d+2}} \cdot \frac{1}{||P||_2^{d-1}}$ où $||P||_2 = \sqrt{\sum_{i=0}^{d} a_i^2}$ (la encore ... il y a mieux).

dans tous les cas : nombre exponentiel d'opérations

Et dieu créa la bissection

On se ramène au cas d'un polynôme dont les racines sont dans [0,1] après calcul de borne et changement de variable

- $k = 0, c = 0, sol = \{\}, aregarder = \{[0,1]\}$
- tant que $pasvide(aregarder)$ zyva
  - $I_{k,c} = [c/2^k, (c+1)/2^k] = unelement(aregarder)$;
    $aregarder = aregarder \setminus I_{k,c}$
  - Si $aexactementunerasin(P, I_{k,c})$, $sol = sol \bigcup \{I_{k,c}\}$
  - Si $aplusdunerasin(P, I_{k,c})$,
    $aregarder = aregarder \bigcup \{I_{k+1,2c}, I_{k+1,2c+1}\}$

En utilisant par exemple les suites de Sturm pour les opérations de décision, $k \leq sep(P)$ et il y a au plus d intervalles à étudier pour k fixé soit au total $O(d^2(t + log_2(d)))$ intervalles à visiter.

Avec Sturm

- On calcule une fois pour toutes la suite de Sturm associée à P et P' ($O(dlog(d)^2)$ avec le meilleur algorithme théorique connu.
- les procedures de décision $aexactementunerasin$ et $aplusdunerasin$ consistent à évaluer toute la suite en les bornes des intervalles considérés (la différences des nombres de changements de signes donnant exactement le nombre de zéros réels dans l'intervalle): $O(d^2)$ opérations arithmétiques (multiplications et additions) à chaque fois avec des coeffs de taille dt (si rationnels).

Le Sturm-Bissection $O(d^4(t + log_2(d))Mul(dt))$ bit opérations et consomme $O(d^3t)$ bits pour le stockage des données.

Remplaçons Sturm par Descartes

Quelques Transformations :

- $R(P(x)) = x^{\deg(P)} P(1/x)$,
- $H_c(P(x)) = P(cx)$ pour $c \in \mathbb{R}$,
- $T_c(P(x)) = P(x+c)$ pour $c \in \mathbb{R}$.

Notation : $P_{k,c} = 2^{nk} P(\frac{x+c}{2^k})$

Remarque : Les racines réelles de $P_{k,c}$ dans $]0,1]$ correspondent aux racines de P dans $I_{k,c} =]c/2^k, (c+1)/2^k]$.

Variante la règle de Descartes :

$$\boxed{Z_{]0,1[}(P_{k,c}) \leq Var(T_1 R(P_{k,c})) \text{ et } Var(T_1 R(P_{k,c})) - Z_{]0,1[}(P_{k,c}) \text{ est pair}}$$

$Z_{]0,1]}(P_{k,c})$ = nombre de racines de $P_{k,c}$ dans $]0,1]$ comptées avec multiplicités = nombre de racines de P dans $I_{k,c}$ comptées avec multiplicités.

Remplaçons Sturm par Descartes

Pourquoi faire ?

- $P_{2k,2c} = 2^n P_{k,c}(X/2)$
- $P_{2k,2c+1} = P_{2k,2c}(X+1)$

On peut montrer que $P(X+1)$ se calcule en $O(n^2)$ additions/soustractions et sans multiplications ni divisions :

for $i = 1$ to n for $k = n - i$ to $n - 1$ $P[k] = P[k] + P[k+1]$;

Evaluer une suite de Sturm coûte $O(n^2)$ multiplications et additions.

opérations sur des entiers de taille t :

addition : linéaire ($O(t)$)

multiplications : naïve: $O(t^2)$, karatsuba : $O(t^{log_2(3)})$

Note : il existe des algorithmes plus perfomants en théorie (fft) mais peu utilsés en pratique.

Pourquoi ça marche

Théorème (Vincent) : $P \in \mathbb{R}[X]$

- sans facteur carré;
- aucune racine complexe dans le disque $D((0, 1/2), 1/2)$

$$\boxed{V(T_1 R(P)) = 0}$$

Théorème (Collins/Johnson) : $P \in \mathbb{R}[X]$

- sans facteur carré;
- une seule racine (réelle) dans $]0, 1[$;
- aucune autre racine (complexe) dans les disques $D((0, 0), 1)$ et $D((1, 0), 1)$.

$$\boxed{V(T_1 R(P)) = 1}$$

Nombre d'intervalles à visiter du même ordre de grandeur que pour Sturm-Bissection mais opérations moins chères

A universal solution : the CAD

Compute (at least) one point on each (semi-algebraically) connected component of a cylindrical decomposition for
all the sign conditions defined by a set of polynomials.

A universal solution : the CAD

Compute (at least) one point on each (semi-algebraically) connected component of a cylindrical decomposition for all the sign conditions defined by a set of polynomials. Powerful (in practice) for quantifier elimination (no *practical* alternative)

A universal solution : the CAD

Compute (at least) one point on each (semi-algebraically) connected component of a cylindrical decomposition for all the sign conditions defined by a set of polynomials. Powerful (in practice) for quantifier elimination (no *practical* alternative)
Inefficient (in practice) for solving zero-dimensional systems (*practical* alternatives)

A universal solution : the CAD

Compute (at least) one point on each (semi-algebraically) connected component of a cylindrical decomposition for all the sign conditions defined by a set of polynomials. Powerful (in practice) for quantifier elimination (no *practical* alternative)
Inefficient (in practice) for solving zero-dimensional systems (*practical* alternatives)

- *solves* more than a system of equalities and inequalities in any case !

A universal solution : the CAD

Compute (at least) one point on each (semi-algebraically) connected component of a cylindrical decomposition for all the sign conditions defined by a set of polynomials. Powerful (in practice) for quantifier elimination (no *practical* alternative)
Inefficient (in practice) for solving zero-dimensional systems (*practical* alternatives)

- *solves* more than a system of equalities and inequalities in any case !
- size of intermediate objects in computations (resultants/sub-resultants);

A universal solution : the CAD

Compute (at least) one point on each (semi-algebraically) connected component of a cylindrical decomposition for all the sign conditions defined by a set of polynomials. Powerful (in practice) for quantifier elimination (no *practical* alternative)
Inefficient (in practice) for solving zero-dimensional systems (*practical* alternatives)

- *solves* more than a system of equalities and inequalities in any case !
- size of intermediate objects in computations (resultants/sub-resultants);
- size of the output (doubly exponential in the number of variables).

CAD - Projection Step

Projection step

At level k, we have a set P_k of polynomial of $K[X_k, \ldots, X_n]$. We construct $P_{k+1} = Proj(P_k)$ as being the smallest set such that :

- If $p \in P_k$, $\deg_{X_k}(p) = d \geq 2$, $Proj(P_k)$ contains all the $sr_j(p, \frac{\partial p}{\partial X_k})$ (non-constant) for $j = 0, \ldots, d$.
- If $p \in P_k$, $q \in P_k$, $Proj(P_k)$ contains $sr_j(p, q)$ (non-constant) for $j = 0, \ldots, \min(\deg_{X_k}(p), \deg_{X_k}(q))$.
- If $p \in P_k$, $\deg_{X_k}(p) \geq 1$ and $lc_{X_k}(p)$ non constant, $Proj(P_k)$ contains $lc_{X_k}(p)$ and $Proj(P_k \setminus \{p\} \cup \{p - lc_{X_k}(p)\})$.
- If $p \in P_k$, $\deg_{X_k}(p) = 0$ and p non constant, $Proj(P_k)$, contains p.

CAD - lifting step

- (1) compute real roots of all polynomials of P_k and sort them;
- (2) take one point on each interval between roots of (1);
- (3) specialize X_k to (1) and (2) in $P_{k-1} \ldots P_1$.

Excepted for $k = n$ step (1) lead to isolate the real roots of polynomial with real algebraic numbers as coefficients which is, in practice, a difficult task.

Also, the basic CAD algorithm can be easily described and implemented using exclusively operations with univariate polynomials. The exponential behavior of the method is mainly due to the projection step and in particular the increase of polynomial degrees due to subresultant computations.

Computable Solutions ?

The best situation is to get a parametrization of the solutions :

Computable Solutions ?

The best situation is to get a parametrization of the solutions :

- difficult in general when the system has an infinite number of (complex) solutions;

Computable Solutions ?

The best situation is to get a parametrization of the solutions :

- difficult in general when the system has an infinite number of (complex) solutions;
- when the system has a finite number of solutions, on may hope to compute something like :

$$f_0(T) = 0, X_1 = f_1(T), \ldots, X_n = f_n(T)$$

with $f_i \in K[T]$.

Computable Solutions ?

The best situation is to get a parametrization of the solutions :

- difficult in general when the system has an infinite number of (complex) solutions;
- when the system has a finite number of solutions, on may hope to compute something like :

$$f_0(T) = 0, X_1 = f_1(T), \ldots, X_n = f_n(T)$$

with $f_i \in K[T]$.

Fundamental question : are two systems equivalents ?

Ideals and Varities

The ideal $\mathcal{I} = \langle P_1, \ldots, P_s \rangle$ generated by $P_1, \ldots, P_s$ is the set of polynomials in $\mathbb{Q}[X_1, \ldots, X_n]$ constituted by all the combinations $\sum_{k=1}^{R} P_k U_k$ with $U_k \in \mathbb{Q}[X_1, \ldots, X_n]$.

$V_{\mathbb{C}}(S) = V_{\mathbb{C}}(I) = \{x \in \mathbb{C}^n \mid p(x) = 0 \; \forall p \in \mathcal{I}\}$ (resp. $V_{\mathbb{R}}(S) = V_{\mathbb{R}}(I) = V_{\mathbb{C}}(I) \bigcap \mathbb{R}^n$) the set of complex (resp. real) zeroes of S.

$$I(V_C(\mathcal{I})) = \sqrt{\mathcal{I}} = \{p \; , \; \exists n \in \mathbb{N} \mid p^n \in \mathcal{I}\}$$

We say that two systems are equivalent iff they generate the same ideal

Monomial Orderings

- In one variable, computations using gcd, resultants etc.

Monomial Orderings

- In one variable, computations using gcd, resultants etc.
- In several variables, one have first to choose an admissible monomial ordering.

Monomial Orderings

- In one variable, computations using gcd, resultants etc.
- In several variables, one have first to choose an admissible monomial ordering.
  - *lexicographic order* : (Lex)

$$X_1^{\alpha_1} \cdot \ldots \cdot X_n^{\alpha_n} <_{Lex} X_1^{\beta_1} \cdot \ldots \cdot X_n^{\beta_n} \Leftrightarrow$$
$$\exists i_0 \leq n \ , \begin{cases} \alpha_i = \beta_i \ , \ \forall i = 1 \ldots i_0 - 1, \\ \alpha_{i_0} < \beta_{i_0} \end{cases}$$

Monomial Orderings

- In one variable, computations using gcd, resultants etc.
- In several variables, one have first to choose an admissible monomial ordering.
  - *lexicographic order* : (Lex)

$$X_1^{\alpha_1} \cdot \ldots \cdot X_n^{\alpha_n} <_{Lex} X_1^{\beta_1} \cdot \ldots \cdot X_n^{\beta_n} \Leftrightarrow$$
$$\exists i_0 \leq n \ , \begin{cases} \alpha_i = \beta_i \ , \ \forall i = 1 \ldots i_0 - 1, \\ \alpha_{i_0} < \beta_{i_0} \end{cases}$$

  - *degree reverse lexicographic order (DRL)* :

$$X_1^{\alpha_1} \cdot \ldots \cdot X_n^{\alpha_n} <_{DRL} X_1^{\beta_1} \cdot \ldots \cdot X_n^{\beta_N}$$
$$\Leftrightarrow X^{\left((\sum_k \alpha_k), \beta_n, \ldots, \beta_1\right)} <_{Lex} X^{\left((\sum_k \beta_k), \alpha_n, \ldots, \alpha_1\right)}$$

Euclidean division

Fixing a monomial ordering induces :

Euclidean division

Fixing a monomial ordering induces :

- $LM(P,<) = max_{i=0...r\ ,<} X^{\mu^{(i)}}$ (leading monomial of P w.r.t. $<$)

Euclidean division

Fixing a monomial ordering induces :

- $LM(P,<) = max_{i=0...r\ ,<} X^{\mu^{(i)}}$ (leading monomial of P w.r.t. $<$)
- $LC(P,<) = a_i$ with i such that $LT(P) = X^{\mu^{(i)}}$ (leading coefficient of P w.r.t. $<$)

Euclidean division

Fixing a monomial ordering induces :

- $LM(P,<) = max_{i=0...r\ ,<} X^{\mu(i)}$ (leading monomial of P w.r.t. $<$)
- $LC(P,<) = a_i$ with i such that $LT(P) = X^{\mu(i)}$ (leading coefficient of P w.r.t. $<$)
- $LT(P,<) = LC(P,<) \cdot LM(P,<)$ (leading term of P w.r.t. $<$)

Euclidean division

Fixing a monomial ordering induces :

- $LM(P,<) = max_{i=0...r\ ,<} X^{\mu^{(i)}}$ (leading monomial of P w.r.t. $<$)
- $LC(P,<) = a_i$ with i such that $LT(P) = X^{\mu^{(i)}}$ (leading coefficient of P w.r.t. $<$)
- $LT(P,<) = LC(P,<) \cdot LM(P,<)$ (leading term of P w.r.t. $<$)

$\Rightarrow$ On can then define a division with respect to one or several polynomials

Euclidean division

Fixing a monomial ordering induces :

- $LM(P,<) = max_{i=0...r\ ,<} X^{\mu(i)}$ (leading monomial of P w.r.t. $<$)
- $LC(P,<) = a_i$ with i such that $LT(P) = X^{\mu(i)}$ (leading coefficient of P w.r.t. $<$)
- $LT(P,<) = LC(P,<) \cdot LM(P,<)$ (leading term of P w.r.t. $<$)

$\Rightarrow$ On can then define a division with respect to one or several polynomials BUT the euclean division do not provide canonical results :

Euclidean division

Fixing a monomial ordering induces :

- $LM(P,<) = max_{i=0...r}{}_{,<} X^{\mu(i)}$ (leading monomial of P w.r.t. $<$)
- $LC(P,<) = a_i$ with i such that $LT(P) = X^{\mu(i)}$ (leading coefficient of P w.r.t. $<$)
- $LT(P,<) = LC(P,<) \cdot LM(P,<)$ (leading term of P w.r.t. $<$)

$\Rightarrow$ On can then define a division with respect to one or several polynomials BUT the euclean division do not provide canonical results :

- (lexicographique ordering with $x > y$) $f = xy^2 - x$, $l = \{g_1 = xy + 1, g_2 = y^2 - 1\}$, Divide($f$,$g_1$,$<$)=$-x - y$, and so Divide($f$,$\{g_1, g_2\}$)=$-x - y$

Euclidean division

Fixing a monomial ordering induces :

- $LM(P,<) = max_{i=0...r}{}_{,<} X^{\mu(i)}$ (leading monomial of P w.r.t. $<$)
- $LC(P,<) = a_i$ with i such that $LT(P) = X^{\mu(i)}$ (leading coefficient of P w.r.t. $<$)
- $LT(P,<) = LC(P,<) \cdot LM(P,<)$ (leading term of P w.r.t. $<$)

$\Rightarrow$ On can then define a division with respect to one or several polynomials BUT the euclean division do not provide canonical results :

- (lexicographique ordering with $x > y$) $f = xy^2 - x$, $l = \{g_1 = xy + 1, g_2 = y^2 - 1\}$, Divide($f$,$g_1$,$<$)=$-x - y$, and so Divide($f$,$\{g_1, g_2\}$)=$-x - y$
- but :$f = xg_2 + 0$.

Representation of ideals : Gröbner bases

A Gröbner base of any polynomial system est an equivalent system : algebraic informations are preserved (multiplicities, dimension, degree).

Representation of ideals : Gröbner bases

A Gröbner base of any polynomial system est an equivalent system : algebraic informations are preserved (multiplicities, dimension, degree).

Input: A set of polynomials S and an admissible monomial ordering $>$.

Representation of ideals : Gröbner bases

A Gröbner base of any polynomial system est an equivalent system : algebraic informations are preserved (multiplicities, dimension, degree).

Input: A set of polynomials S and an admissible monomial ordering $>$.

Output:

- A set of generators G such that $< G >=< S >$;

Representation of ideals : Gröbner bases

A Gröbner base of any polynomial system est an equivalent system : algebraic informations are preserved (multiplicities, dimension, degree).

Input: A set of polynomials S and an admissible monomial ordering $>$.

Output:

- A set of generators G such that $< G >=< S >$;
- A *normalform* (NF) function say :

Representation of ideals : Gröbner bases

A Gröbner base of any polynomial system est an equivalent system : algebraic informations are preserved (multiplicities, dimension, degree).

Input: A set of polynomials S and an admissible monomial ordering $>$.

Output:

- A set of generators G such that $< G >=< S >$;
- A *normalform* (NF) function say :
  - $NF(p, G, >) = p$ modulo $< S >$;

Representation of ideals : Gröbner bases

A Gröbner base of any polynomial system est an equivalent system : algebraic informations are preserved (multiplicities, dimension, degree).

Input: A set of polynomials S and an admissible monomial ordering $>$.

Output:

- A set of generators G such that $< G >=< S >$;
- A *normalform* (NF) function say :
  - $NF(p, G, >) = p$ modulo $< S >$;
  - $NF(p, G, >) = 0$ iff $p \in < S >$.

Representation of ideals : Gröbner bases

A Gröbner base of any polynomial system est an equivalent system : algebraic informations are preserved (multiplicities, dimension, degree).

Input: A set of polynomials S and an admissible monomial ordering $>$.

Output:

- A set of generators G such that $< G >=< S >$;
- A *normalform* (NF) function say :
  - $NF(p, G, >) = p$ modulo $< S >$;
  - $NF(p, G, >) = 0$ iff $p \in < S >$.
  - $NF(p, G, >)$ uniquely defined;

Degree / Dimension

- dimension = number of degrees of freedom = dimension of the variety;

Degree / Dimension

- dimension = number of degrees of freedom = dimension of the variety;
- degree = maximum number of solutions *counted with multiplicities* once free variables are generically set $\neq$ degree of the variety;

Degree / Dimension

- dimension = number of degrees of freedom = dimension of the variety;
- degree = maximum number of solutions *counted with multiplicities* once free variables are generically set $\neq$ degree of the variety;

the *algebraic* degree and the dimension can easily be computed from a Gröbner basis (hilbert function).

Lexicographic Gröbner bases

$$
\left\{
\begin{array}{l}
f_{1,1}(X_1) \\
f_{2,1}(X_2, X_1) \\
\ldots \\
f_{2,n_2}(X_2, X_1) \\
\hline
\ldots \\
\hline
f_{i,1}(X_i, X_{i-1}, \ldots, X_1) \\
\ldots \\
f_{i,n_i}(X_i, X_{i-1}, \ldots, X_1) \\
\hline
\ldots \\
\hline
f_{n,1}(X_n, X_{n-1}, \ldots, X_1) \\
\ldots \\
f_{n,n_n}(X_n, X_{n-1}, \ldots, X_1)
\end{array}
\right.
$$

Extraction of triangulat sets

$$
\left\{
\begin{array}{l}
f_{1,1}(X_1) \\
f_{2,1}(X_2, X_1) \\
\ldots \\
f_{2,n_2}(X_2, X_1) \\
\hline
\ldots \\
\hline
f_{i,1}(X_i, X_{i-1}, \ldots, X_1) \\
\ldots \\
f_{i,n_i}(X_i, X_{i-1}, \ldots, X_1) \\
\hline
\ldots \\
\hline
f_{n,1}(X_n, X_{n-1}, \ldots, X_1) \\
\ldots \\
f_{n,n_n}(X_n, X_{n-1}, \ldots, X_1)
\end{array}
\right.
$$

Triangular Sets

$$\begin{cases} f_{1,1}(X_1) \\ f_{2,1}(X_2, X_1) \\ \dots \\ f_{j,i}(X_i, X_{i-1}, \dots, X_1) \\ \dots \\ t_{j,n}(X_n, X_{n-1}, \dots, X_1) \end{cases}$$

A first property :

- $W(T) = V(T) \setminus V(\prod_{i=d+1}^{n} LC(t_i, X_i))$;
- if $< S >$ is prime and $W(T) \neq 0$ then $Adh(W(T)) = V(S)$;

Important Property of triangulat sets

Regular triangular set : $LC(t_i, X_i) \notin$ any prime ideal associated to $sat(t_{d+1}, \ldots, t_i) \bigcap K[X_1, \ldots, X_i]$.

Separable triangular set : $\frac{\partial t_i}{\partial X_i} \notin$ any prime ideal associated to $sat(t_{d+1}, \ldots, t_i) \bigcap K[X_1, \ldots, X_i]$.

with :

$$sat(T) = \{p \in K[X_1, \ldots, X_n] \ , \ \exists n \in N \ , \ h \in < lc(t_i, X_i)) > \ , \ h^m p \in < T >\}$$

Theorem [ALM]: if T is extracted from G (lexicographic Gröbner basis) , if $sat(T) =< G >$ and if T is regular and separable, then $< G >$ is radical and equidimensional.

Gröbner bases and localization

S a system, P any polynomial, T a new wariable, and G_T a lexicographic Gröbner base of $S, TP - 1$ (with $T > X_1, \ldots X_n$).

If we remove from G_T all the polynomials depending on T, we obtain a Gröbner basis G_L defining the smallest variety that contains $V_C(S) \setminus \{x\mathbb{C}^n \ , \ p(x) = 0\}$.

In particular : if $G_L = G$, then $p \notin$ any prime ideal associated to G.

Equidimensional and Radical decomposition

theorem [ALM] gives an algorithm for computing an equidimensional decomposition :

Input : a lexicographic Gröbner basis G.

Output : a set of couples (G_i, T_i) such that :

- G_i are lexicographic Gröbner bases;
- T_i are regular and separable triangular sets;
- $sat(T_i) =< G_i >$;
- $V(G) = \bigcup V(G_i)$.

Zero-dimensional Systems

One can read on a Gröbner basis if the system has a finite number of complex roots :

Let S any algebraic system and G a Gröbner base of S for any admissible ordering $<$. S has a finite number of real roots iff $\forall i = 1 \ldots n\ ,\ \exists g \in G\ n_i \in \mathbb{N} \mid LM(g, <) = X_i^{n_i}$

In fact, $Irr = \{m \in M[X_1, \ldots, X_n]\ ,\ NF(m, G, <) = m\}$ is a basis of the finite dimensional K-vector-space $\frac{K[X_1,\ldots,X_n]}{<S>}$, and $\sharp V_C(< S >) = \sharp Irr$.

Stickelberger Theorem

q any polynomial in $K[X_1, \ldots, X_n]$

$$
\begin{array}{rccc}
m_q : & \frac{K[X_1,\ldots,X_n]}{<S>} & \longrightarrow & \frac{K[X_1,\ldots,X_n]}{<S>} \\
 & NF(p, G, <) & \longmapsto & NF(pq, G, <)
\end{array}
$$

The eigenvalues of m_q are exactly the $q(\alpha)$ where $\alpha \in V_{\mathbb{C}}(S)$.

Lexicographic Gröbner bases

In the zero-dimensional case , a lexicographic Gröbner basis can be deduced from any other Gröbner basis using exclusively linear algebra tehcnics;

basic principle : change the base of $\frac{K[X_1,...,X_n]}{<S>}$

Shape Lemma

If the complex roots have distinct first coordinates and if the system has no multiplicity, then a lexicographic Gröbner basis for $X_1 < X_2 < \ldots < X_n$ has always the following shape :

$$\begin{cases} f(X_1) \\ X_2 - f_2(X_1) \\ \vdots \\ X_n - f_n(X_1) \end{cases}$$

Rational Univariate Representation

From any Gröbner basis of $< S >$, one can compute a rational parametrization of all the roots of the system :

$$\begin{cases} f(T) = 0 \\ X_1 = g_1(T)/g(T) \\ \vdots \\ X_n = g_n(T)/g(T) \end{cases}$$

which induces a bijection between $V_C(< S >)$ and $V_C(< f >)$ that preserves the real roots and the multipicities.

Utilisation d'arithmétiques d'intervalles

Objectifs :

- accélérer les calculs;
- traiter certains problèmes où les coefficients ne sont pas connus exactement;
- traiter certains problèmes où les coefficients sont des nombres algébriques réels.

Problèmes :

- gérer l'instabilité numérique;
- gérer le *déplacement* des racines;
- obtenir un résultat exact.

Quelques remarques

Besoin d'une faible précision :

Les algos basés sur la règle de Descartes :

1. calculent un polynome $P_{k,c} = P(\frac{x+c}{2^k})$;
2. regardent $V(T_1R(P_{c,k}))$.

seuls les signes des coefficients de $T_1R(P_{c,k})$ sont importants.

Seul le nombre de variations de signes de $T_1R(P_{c,k})$ est essentiel

Exemple 1 : $V(]1.1, 1.2] * X^2 +] - 0.1; 0.1] * X +] - 1.2, -1.1]) = 1$

Exemple 2 : $V(]1.1, 1.2] * X^2 +]1.3; 1.4] * X +] - 0.1, 0.1]) = 1$ ou 0

Quelques remarques (2)

Règles :

- 2 signes indeterminés de suite $\Rightarrow$ on ne peut pas déduire le nombre de variations de signes.
- dans les situations $]..,+,?,+,..]$ et $]..,-,?,-,..]$, on ne peut pas déduire le nombre de variations de signes.
- dans les situations $]-,?,+]$ et $]+,?,-]$, le nombre de variations de signes est toujours 1.

Problème : Les méthodes basées sur la règle de Descartes ne sont correctes que si le polynome considéré est **sans facteur carré.**

Exemple 3 : $]0.9,1.1]*X^2+]-2.1,-1.9]*X+]0.9,1.1]$ *contient* $(X-1)^2$.

Spécifications de l'algorithme utilisant une arithmétique d'intervalles

`DescartesBoundInterv(`P`)`

Idem que `DescartesBound(`P`)` mais générant une erreur (`usp_error`) si la décision sur le nombre de changements de signes n'est pas possible.

Descartes et intervalles

GenericDescartesInterv

Input: Idem `GenericDescartes`

Output: $E = \texttt{Exact}(P)$ $I = \texttt{Isol}(P)$ et $\texttt{Error}(P)$.

Auxiliary function: $\texttt{DescartesBoundInterv}(P)$.

- $E \leftarrow \emptyset, \quad I \leftarrow \emptyset, \quad T \leftarrow \texttt{initializeTree}(P)$
- while $T \neq \emptyset$ do
  - $(k, c, Q) \leftarrow \texttt{getNewNode}(T, <)$
  - $s \leftarrow \texttt{DescartesBoundInterv}(Q)$
  - if (`usp_error`) $\texttt{Error}(P) \leftarrow \texttt{Error}(P) \cup \{(k, c)\}$
  - else
    - if $s = 1$ then $I \leftarrow I \cup \{(k, c)\}$
    - if $s > 1$ then $(E, T) \leftarrow \texttt{addSuccessors}((k, c), Q, E, T)$

Terminaison de `GenericDescartesInterv`

Lemme :
Si $\exists (b_0, \ldots, b_n) \in \prod_{i=0}^{d} \lfloor a_i \rceil$ tq $\sum_{i=0}^{n} b_i X^i$ sans facteur carré, alors `DescartesHyb` termine.

Lemme :
Si $\exists i \in [0, \ldots, n]$ tq $Largeur(\lfloor a_i \rceil) > 0$, alors $\exists (b_0, \ldots, b_n) \in \prod_{i=0}^{d} \lfloor a_i \rceil$ tq $\sum_{i=0}^{n} b_i X^i$ sans facteur carré.

Proposition : Si les coefficients de P ne sont pas connus exactement,

`GenericDescartesInterv`(P) termine toujours !

Tous comptes faits ...

ReduceIfCan(P)

Input : $P = \sum_{i=0}^{n} \lfloor a_i \rceil$

Output : $Q = \sum_{i=0}^{n} \lfloor c_i \rceil$ with the same roots than P and such that $\exists (b_0, \ldots, b_n) \in \prod_{i=0}^{d} \lfloor c_i \rceil$ such that $\sum_{i=0}^{n} b_i X^i$ is square-free.

- if ($Largeur(\lfloor a_i \rceil) = 0 \; \forall i \in [0, \ldots, n]$) return($\frac{P}{gcd(P, P')}$)
- else return(P).

Théorème : `GenericDescartesInterv(ReduceIfCan(`P`))` termine toujours.

Algorithme hybride : cas de polynômes à coefficients exacts

on peut avoir la précision que l'on veut sur l'input.

Algorithme Hybride

GenericDescartesHyb(P)

Input : $P = \sum_{i=0}^{n} a_i X^i \in \mathbb{Z}[X]$

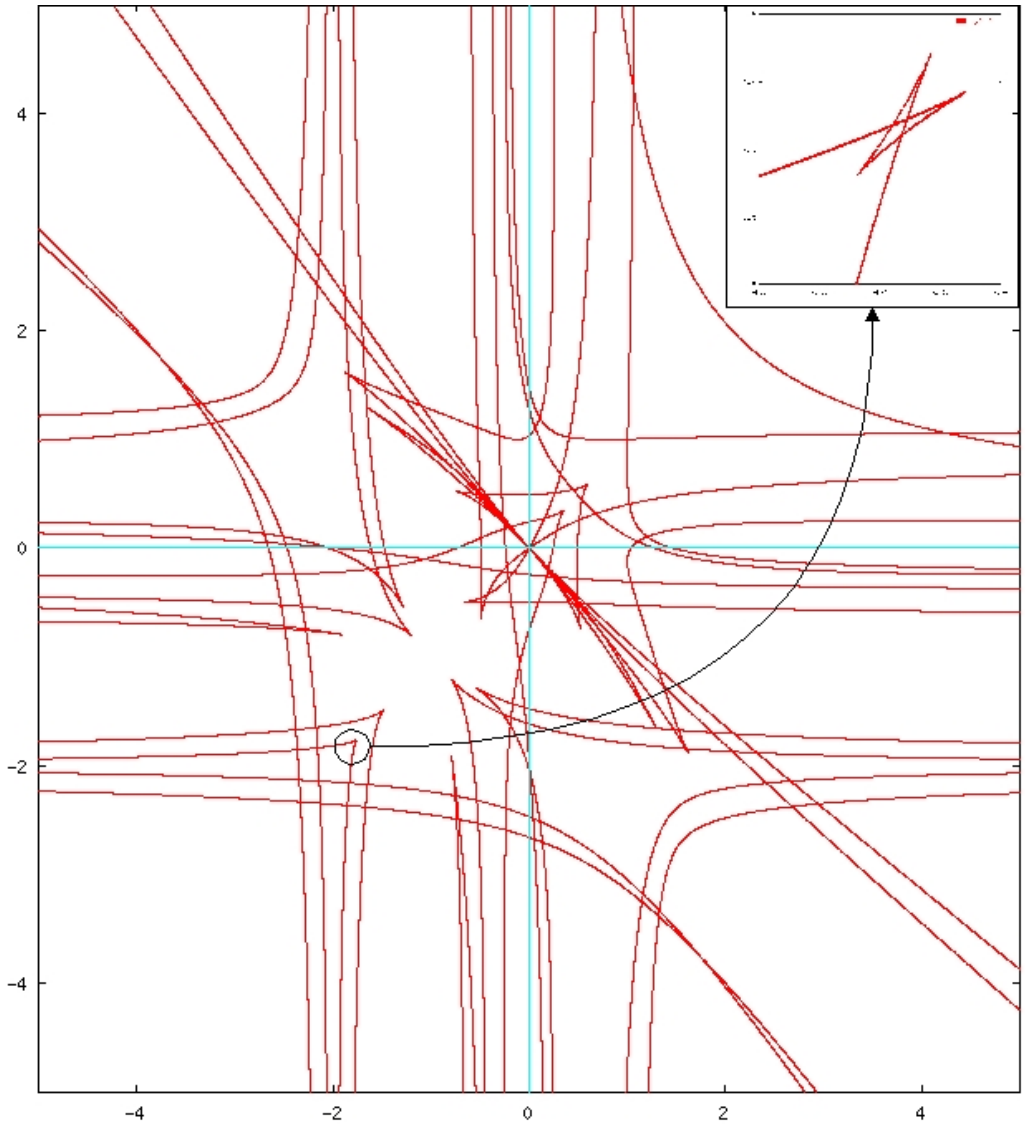
Output : Isol(P) et Exact(P)

$todo =]0, 1]$;

While ($todo \neq \emptyset$)

- $]\frac{c}{2^k}, \frac{c+1}{2^k}] = First(todo)$; remove $]\frac{c}{2^k}, \frac{c+1}{2^k}]$ from $todo$;
- if ($increase(floatPrecision) < maxPrecision$)
  - $Q = ReduceIfCan(convertInterval(P, floatPrecision))$;
  - GenericDescartesInterv($T_c H_{1/2^k}(Q)$,l_{isole},l_{error});
- else GenericDescartes($T_c H_{1/2^k}(P)$,l_{isole}); $l_{error} = \emptyset$;
- Isol(P) = Isol$(P) \cup \{H_{2^k} T_{-c}(I)\ I \in l_{isole}\}$
- $todo = \{H_{2^k} T_{-c}(I)\ I \in l_{error}\}$

En pratique : une courbe degré 424



Expériences

Name	53	107	215	431	863	Exact
Lag 400	0	0	0	7	397	0
Che 500	0	0	0	0	400	0
Wil 400	0	0	0	0	400	0
Mig 200	1	0	0	3	0	0
Mig 300	1	0	0	3	0	0
Fab24	22	2	0	0	0	0
Cyc7	56	0	0	0	0	0
Kat7	20	24	0	0	0	0
Kat8	1	56	27	0	0	0
Kat9	4	16	64	0	0	0
Reimer5	0	0	24	0	0	0
Virasoro	1	39	38	0	138	8

Expériences

Name	MPS	?	OK	Exact	Hyb	Nb
Lag 200	57.4	0	200	3.1	**3.0**	200
Che 500	920.4	0	500	**75.2**	76.2	500
Wil 400	3462.5	0	400	37.9	**35.9**	400
Fab24	0.06	0	24	0.19	**0.04**	24
Cyc7	26.5	0	56	236.1	**21.1**	56
Kat7	0.9	0	44	**0.36**	0.40	44
Kat8	9.1	0	84	**4.41**	7.63	84
Kat9	74.3	0	120	79.8	**57.02**	120
Reimer5	8.8	0	24	**0.32**	0.5	24
Virasoro	147.5	0	224	**17.2**	**17.2**	224

Polynômes dont les coeffs sont des intervalles

On ne peut pas avoir la précision que l'on veut sur l'input.

Problème : valeurs des coeffs ou le polynôme n'est pas sans facteurs carrés : résultat incomplet mais exact !

Polynômes dont les coeffs sont algébriques réels

- N.A.R représenté par 1 intervalle + une méthode pour le raffiner; application directe de l'algorithme hybride : résultat évent. incomplet. On peut imaginer utiliser des bornes de séparation pour assurer les tests de signes : comportement désastreux dans certains cas.
- $P(T, X) = 0$ et $Q(T) = 0$
  - méthode bestiale : calculer une RUR du système. Degré des systèmes !
  - application directe de l'algorithme général en considerant $P \in \mathbb{Z}[T][X]$. Le calcul du signe des coefficients pouvant se faire en appliquant récursivement l'algorithme de Uspensky et en calculant des gcd ... peut convenir si une seule extension
  - Assurer que P est sans facteurs carrés puis appliquer la méthode hybride. Factorisation absolue ou de façon plus légère, ensembles triangulaires régulier séparables. Attention au raffinement

En pratique

Sur des exemples de calcul d'intersection de quadriques (après paramétrisation des solutions)

$P_1(X_1) = 0, P_2(X_2) = 0, P_3(X_3) = 0, P(T, X_1, X_2, X_3) = 0$ avec P_i de degré 2 et P de degré 8.

1 calcul = plusieurs milliers de systèmes de ce genre.

Stratégie : Intervalles (doubles) puis Ens. Tri. avec rafinement bloqué ($1/2^{32}$) si échec, enfin RUR si toujours pas de réponse.

Bilan : en moyenne, sur des exemples pratiques, pour 5000 calculs, une dizaine ne passent pas avec le Descartes/intervalles, et très rares sont les cas ou un calcul de RUR est nécessaire.

Frequently asked questions

- (1) Emptiness Real Algebraic Sets / one point;

Frequently asked questions

- (1) Emptiness Real Algebraic Sets / one point;
- (2) Emptiness of Real Semi-Algebraic Sets/one point w.r.t. sign conditions;

Frequently asked questions

- (1) Emptiness Real Algebraic Sets / one point;
- (2) Emptiness of Real Semi-Algebraic Sets/one point w.r.t. sign conditions;
- (3) Quantifier Elimination;

Frequently asked questions

- (1) Emptiness Real Algebraic Sets / one point;
- (2) Emptiness of Real Semi-Algebraic Sets/one point w.r.t. sign conditions;
- (3) Quantifier Elimination;

Frequently asked questions

- (1) Emptiness Real Algebraic Sets / one point;
- (2) Emptiness of Real Semi-Algebraic Sets/one point w.r.t. sign conditions;
- (3) Quantifier Elimination;

(1') One point on each semi-algebraically connected component of a real algebraic set

Relation between (1') and more general problems

One point on each semi-algebraically connected component of

$$X = \{x \in \mathbb{R}^n | P_1(x) = \ldots P_l(x) = 0, P_{l+1}(x) > 0, \ldots, P_s(x) > 0\}$$

Proposition : If C is a cell of a non empty sign condition of X, we can find an algebraic set $V \in (R(\epsilon))^k$ defined by :

$$P_1 = \ldots P_l = 0, P_{i_1} - \epsilon = 0, \ldots, P_{i_m} - \epsilon = 0$$

such that a cell of V is contained in $C_{R<\epsilon>}$.

tutorial : M.F. Roy - basic algorithms in real algebraic geometry ...

Critical Points methods

Critical Points methods

Compute (at least) one point on each (semi-algebraically) connected component of a real algebraic set

Critical Points methods

Compute (at least) one point on each (semi-algebraically) connected component of a real algebraic set

The basic objective : Avoid cascading projections.

Critical Points methods

Compute (at least) one point on each (semi-algebraically) connected component of a real algebraic set

The basic objective : Avoid cascading projections.

The main idea : Use Morse functions with a finite number of critical points and reduce the problem to the study of zero-dimensional systems with a single exponential (w.r.t. n) number of solutions.

Critical Points methods

Compute (at least) one point on each (semi-algebraically) connected component of a real algebraic set

The basic objective : Avoid cascading projections.

The main idea : Use Morse functions with a finite number of critical points and reduce the problem to the study of zero-dimensional systems with a single exponential (w.r.t. n) number of solutions.

$\Rightarrow$Resolution of zero-dimensional systems a basic black-box.

Critical Points methods

Compute (at least) one point on each (semi-algebraically) connected component of a real algebraic set

The basic objective : Avoid cascading projections.

The main idea : Use Morse functions with a finite number of critical points and reduce the problem to the study of zero-dimensional systems with a single exponential (w.r.t. n) number of solutions.

$\Rightarrow$Resolution of zero-dimensional systems a basic black-box.

f **= coordinate function.**

f **= Distance function to one point** A**.**

Critical Points methods

Compute (at least) one point on each (semi-algebraically) connected component of a real algebraic set

The basic objective : Avoid cascading projections.

The main idea : Use Morse functions with a finite number of critical points and reduce the problem to the study of zero-dimensional systems with a single exponential (w.r.t. n) number of solutions.

$\Rightarrow$Resolution of zero-dimensional systems a basic black-box.

f **= coordinate function.** If V is compact and smooth,

f **= Distance function to one point** A**.**

Critical Points methods

Compute (at least) one point on each (semi-algebraically) connected component of a real algebraic set

The basic objective : Avoid cascading projections.

The main idea : Use Morse functions with a finite number of critical points and reduce the problem to the study of zero-dimensional systems with a single exponential (w.r.t. n) number of solutions.

$\Rightarrow$Resolution of zero-dimensional systems a basic black-box.

f **= coordinate function.** If V is compact and smooth, the set of critical points of f is finite and intersects every semi-algebraically connected components.

f **= Distance function to one point** A**.**

Critical Points methods

Compute (at least) one point on each (semi-algebraically) connected component of a real algebraic set

The basic objective : Avoid cascading projections.

The main idea : Use Morse functions with a finite number of critical points and reduce the problem to the study of zero-dimensional systems with a single exponential (w.r.t. n) number of solutions.

$\Rightarrow$Resolution of zero-dimensional systems a basic black-box.

f **= coordinate function.** If V is compact and smooth, the set of critical points of f is finite and intersects every semi-algebraically connected components.

f **= Distance function to one point** A**.** If V is smooth and A generic enough,

Critical Points methods

Compute (at least) one point on each (semi-algebraically) connected component of a real algebraic set

The basic objective : Avoid cascading projections.

The main idea : Use Morse functions with a finite number of critical points and reduce the problem to the study of zero-dimensional systems with a single exponential (w.r.t. n) number of solutions.

$\Rightarrow$Resolution of zero-dimensional systems a basic black-box.

f **= coordinate function.** If V is compact and smooth, the set of critical points of f is finite and intersects every semi-algebraically connected components.

f **= Distance function to one point** A**.** If V is smooth and A generic enough, the set of critical points of f is finite and intersects every semi-algebraically connected components.

Choice of the function

Most of the proposed variants of the critical points method use a **coordinate function** or the **distance function** to one point.

Choice of the function

Most of the proposed variants of the critical points method use a **coordinate function** or the **distance function** to one point.

Coordinate function. If V is compact and smooth, the set of critical points is finite and intersects every semi-algebraically connected components.

Choice of the function

Most of the proposed variants of the critical points method use a **coordinate function** or the **distance function** to one point.

Coordinate function. If V is compact and smooth, the set of critical points is finite and intersects every semi-algebraically connected components.

Distance function to one point A. If V is smooth and A generic enough, the set of critical points is finite and intersects every semi-algebraically connected components.

Choice of the function

Most of the proposed variants of the critical points method use a **coordinate function** or the **distance function** to one point.

Coordinate function. If V is compact and smooth, the set of critical points is finite and intersects every semi-algebraically connected components.

Distance function to one point A. If V is smooth and A generic enough, the set of critical points is finite and intersects every semi-algebraically connected components.

$\Rightarrow$Distance function (Coordinate function under study by M. Safey El Din and E. Schost).

Algebraic formulation

$V = V(P_1, \ldots, P_s)$ an algebraic variety of dimension d and $A \in \mathbb{Q}^n$,

$$\mathcal{C}(V, A) = \{M \in V, \mathrm{rank}(grad_M(P_1), \ldots, grad_M(P_s), \overrightarrow{AM}) \leq n - d\}.$$

Algebraic formulation

$V = V(P_1, \ldots, P_s)$ an algebraic variety of dimension d and $A \in \mathbb{Q}^n$,

$$\mathcal{C}(V, A) = \{M \in V, \mathrm{rank}(grad_M(P_1), \ldots, grad_M(P_s), \overrightarrow{AM}) \leq n - d\}.$$

Algebraic formulation

$V = V(P_1, \ldots, P_s)$ an algebraic variety of dimension d and $A \in \mathbb{Q}^n$,

$$\mathcal{C}(V, A) = \{M \in V, \mathrm{rank}(grad_M(P_1), \ldots, grad_M(P_s), \overrightarrow{AM}) \leq n - d\}.$$

$\Rightarrow$A priori of smaller dimension than V.

$\langle P_1, \ldots, P_s \rangle$ **need to be at least radical :**

$$V = V(X_2^2) \Rightarrow \mathcal{C}(V, A) = V$$

$\mathcal{C}(V, A)$ has the same dimension than V

Algebraic formulation

$V = V(P_1, \ldots, P_s)$ an algebraic variety of dimension d and $A \in \mathbb{Q}^n$,

$$\mathcal{C}(V, A) = \{M \in V, \mathrm{rank}(grad_M(P_1), \ldots, grad_M(P_s), \overrightarrow{AM}) \leq n - d\}.$$

$\Rightarrow$A priori of smaller dimension than V.

$\langle P_1, \ldots, P_s \rangle$ **need to be at least radical :**

$$V = V(X_2^2) \Rightarrow \mathcal{C}(V, A) = V$$

$\mathcal{C}(V, A)$ has the same dimension than V

$\Rightarrow$May contain the critical points of $d(A, V)$;

Bad examples

$\langle P_1, \ldots, P_s \rangle$ **radical is not sufficient :**

$$P_1 = (X_1^2 + X_2^2 - 1)(X_1 - 2)\ ,\ P_2 = (X_1 - 2)X_3$$

$\mathcal{C}(V, A)$ doesn't meet every semi-algebraically connected component (not equi-dimensional - miss the component of lowest dimension).

Bad examples

$\langle P_1, \ldots, P_s \rangle$ **radical is not sufficient :**

$$P_1 = (X_1^2 + X_2^2 - 1)(X_1 - 2) \ , \ P_2 = (X_1 - 2)X_3$$

$\mathcal{C}(V, A)$ doesn't meet every semi-algebraically connected component (not equi-dimensional - miss the component of lowest dimension).

The condition $\langle P_1, \ldots, P_s \rangle$ **prime is sufficient** if A **is generic enough** (not , for example the center of a circle included in V).

A algorithmic result

$V(P_1, \ldots, P_s)$ an **equidimensional** algebraic variety of dimension d and $\langle P_1, \ldots, P_s \rangle$ **radical.** $A \in K^n$,

$$\mathcal{C}(V, A) = \{M \in V, \mathrm{rank}(grad_M(P_1), \ldots, grad_M(P_s), \overrightarrow{AM}) \leq n - d\}.$$

If D is a positive integer large enough, $\exists A \in \{1 \ldots D\}^n$ such that :

1. $\mathcal{C}(V, A)$ meets every connected component of $V \bigcap \mathbb{R}^n$,
2. $\mathcal{C}(V, A) = \mathrm{Sing}(V) \bigcup V_0$.

where

- V_0 is a finite set of points in $\mathbb{C}^n$,
- $\mathrm{Sing}(V) = \{M \in V \mid \mathrm{rank}(grad_M(P_1), \ldots, grad_M(P_s)) < n - d\}$.

Moreover, $\dim(\mathcal{C}(V, A)) < \dim(V)$.

A Basic Algorithm

- **Input:** A polynomial system S of equations in $K[X_1, \ldots, X_n]$.
- **Output:** A list of zero-dimensional systems whose zero sets intersects every semi-algebraically connected component of $V(S) \bigcap R^n$.

1. $\mathrm{list} := \mathrm{EquiDimDec}(S)$, $\mathrm{result} := \emptyset$,
2. Choose $A \notin V(S)$.
3. while $\mathrm{list} \neq \emptyset$ do
   - $S := \mathrm{first}(\mathrm{list})$, and remove S from list, set $d := \mathrm{Dim}(S)$,
   - if $d = 0$ then $\mathrm{result} := \mathrm{result} \bigcup S$,
   - else
     - (*) $Q := \mathrm{Minors}(S, d, A) \bigcup S$ and set $u := \mathrm{Dim}(Q)$,
     - if $u = d$ choose another point $A \notin V(S)$ and go to step (*),
     - $\mathrm{list} := \mathrm{list} \bigcup \mathrm{EquiDimDec}(Q)$,
4. return result.

Tests

System	Dimension	Degree	Nb. Vars	Output	ARS	QEPCAD
Vermeer	1	26	5	84	3.3	43
Wang	1	114	13	132	1.4	∞
Euler	3	2	10	10	< 1	failed(*872043*)
Neural	1	24	4	133	< 1	< 1
Butcher	3	3	8	15	1.7	∞
Buchberger	4	6	8	32	< 1	failed(*991324*)
DiscPb	2	3	4	28	< 1	∞
Donati	1	10	4	61	10	< 1

Résumé des épisodes précédents

Caractérisation des robots cuspidaux :

$$P = aX^4 + bX^3 + cX^2 + dX + e$$

avec :

$$\begin{cases} a = & m5 - m2 + m0 \\ b = & -2 * m3 + 2 * m1 \\ c = & -2 * m5 + 4 * m4 + 2 * m0 \\ d = & 2 * m1 + 2 * m3 \\ e = & m5 + m2 + m0 \end{cases}$$

$$\begin{cases} m0 = & -R + Z + r_2^2 + (K - L)^2/4 \\ m1 = & 2r_2 d_4 + (L - K) d_4 r_2 \\ m2 = & (L - K) d_4 d_3 \\ m3 = & 2r_2 d_3 d_4^2 \\ m4 = & d_4^2 (r_2^2 + 1) \\ m5 = & d_4^2 d_3^2 \end{cases}$$

$$Z = z^2$$

$$R = x^2 + y^2 + z^2$$

admet-il des points triples ?

Résolution directe

Le système

$$S_0 = \{P, \frac{\partial P}{\partial X}, \frac{\partial^2 P}{\partial X^2}\}$$

est très difficile à résoudre directement. On peut cependant en calculer une base de Gröbner (algorithme F4).

- résultat énorme
- contient beaucoup de composantes inutiles (ex : $d3 <= 0$)
- contient des solutions dégénérées inutiles en pratique (impossible de construire en pratique un robot dans ces composantes compte tenu des imprécisions dues à la fabrication des pièces).

Idée

Trouver la composante générique (ensemble de solutions de dimension 3) dont il faudra enlever les points “dégénérés” (ensembles de solutions de dimension 2).
S’aider en permanence du fait que l’on cherche des solutions réelles pour des valeurs positives des paramètres.
Ce que l’on peut espérer : Une partition de l’ensemble des paramètres (surfaces) telle qu’au dessus de chaque cellule, le nombre de solutions du système initial soit constant + une expression littérale de ces solutions.

Un changement de variable

But : aider l'elimination des variables.

$$\begin{cases} X1 = & e - a & (R, d3, d4, r2) \\ X2 = & d - b & (d3, d4, r2) \\ X3 = & X1 + 2 * a - c & (d3, d4, r2) \\ t = & r2/d3 & (d3, d4, r2) \end{cases}$$

$$S_1 = \{P, \frac{\partial P}{\partial X}, \frac{\partial^2 P}{\partial X^2}\} \in \mathbb{Q}[X, a, X1, X2, X3, t]$$

But : éliminer X et a pour obtenir une equation en $X1$ (ne dépendant que de R et des paramètres $d3, d4, r2$.

Résolution en X et a

$$S_1 = \{P, \frac{\partial P}{\partial X}, \frac{\partial^2 P}{\partial X^2}\} \in \mathbb{Q}[X, a, X1, X2, X3, t]$$

Parmi les polynômes de la base de Gröbner de S_1 pour l'ordre lexicographique $X > a > X1 > X2 > X3 > X1$ on remarque

$$\begin{cases} p_x = & l_x(X2, X3, t)X + c_x(a, X1, X2, X3, t) \\ p_a = & l_a(X2, X3, t)a + c_a(X1, X2, X3, t) \end{cases}$$

Ainsi, si $l_x \neq 0$ et $l_a \neq 0$, on a une expression formelle des coordonnées X et a (c'est à dire Z) en fonction de $R, d3, d4, r2$:

$$\begin{cases} X = & -\frac{c_x(a,X1,X2,X3,t)}{l_x(X2,X3,t)} \\ a = & -\frac{c_a(X1,X2,X3,t)}{l_a(X2,X3,t)} \end{cases}$$

La surface à étudier

Après substitution des solutions formelles en X et a, on obtient une unique équation S_R en $X1, X2, X3, t$, ce qui après substitution donne une équation de degré 6 en R dans $\mathbb{Q}[R, d4, d3, r2]$.

Au dessus de cette surface, les solutions du système en X et Z sont uniquement déterminées dès lors que $l_x \neq 0$ et $l_a \neq 0$.

Il faudra tout de même restreindre l'ensemble des solutions car il faut $Z = z^2 \geq 0$ et $x^2 + y^2 = R - Z \geq 0$.

Etude de la surface

L'équation $S_R = 0$ admet un nombre fini de solutions en dehors des points annulant son discriminant ou con coefficient de plus haut degré.
Ces 2 conditions donnent 2 polynomes de $\mathbb{Q}[d4, d3, r2]$: dis et lc, partitionnant l'espace des paramètres de telle sorte que le nombre de solutions de $S_R = 0$ soit constant au dessus de chaque cellule.
Les equations à rajouter pour obtenir une decomposition similaire mais telle que le nombre de solutions du système initial soit constant au dessus de chaque cellule sont : $l_x = 0$ et $l_a = 0$ (2 polynomes de $\mathbb{Q}[d4, d3, r2]$).

Enfin, rajouter des conditions suffisantes sur les paramètres pour que $Z = 0$ ou $R - Z = 0$ afin de terminer la décomposition en cellules de l'espace des paramètres telle que le nombre de solutions admissibles du probleme initial soit constant au dessus de chaque cellule.

Partition de l’espace des paramètres

Une CAD trafiquée pour n’obtenir que la description des cellules de plus grande dimension.

(uniquement, terme de tete+ discriminant pour chaque polynome, et resultant pour tout couple de polynômes)